#9

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
PAU ET AL.

Serial No. **Not Yet Assigned**

Filing Date: **Herewith**

For: **METHOD AND SCALABLE ARCHITECTURE FOR PARALLEL CALCULATION OF THE DCT OF BLOCKS OF PIXELS OF DIFFERENT SIZES AND COMPRESSION THROUGH FRACTAL CODING**

TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Transmitted herewith is a certified copy of the priority European Application No. 98830522.3.

Respectfully submitted,

CHRISTOPHER F. REGAN
Reg. No. 34,906
Allen, Dyer, Doppelt, Milbrath & Gilchrist, P.A.
255 S. Orange Avenue, Suite 1401
Post Office Box 3791
Orlando, Florida 32802
Telephone: 407/841-2330
Fax: 407/841-2343
Attorney for Applicants

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

"EXPRESS MAIL" MAILING LABEL NUMBER EL434616346US

DATE OF DEPOSIT 9/3/99

I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 CFR 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO THE COMMISSIONER OF PATENTS AND TRADEMARKS, WASHINGTON, D.C. 20231

Eric Link
(TYPED OR PRINTED NAME OF PERSON MAILING PAPER OR FEE)

(SIGNATURE OF PERSON MAILING PAPER OR FEE)

Europäisches Patentamt
European Patent Office
Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

98830522.3

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

Arlette Fiedler

A. Fiedler

Europäisches Patentamt | European Patent Office | Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.: 98830522.3
Demande n°:

Anmeldetag:
Date of filing: 07/09/98
Date de dépôt:

Anmelder:
Applicant(s):
Demandeur(s):
STMicroelectronics S.r.l.
20041 Agrate Brianza (Milano)
ITALY

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Fractal coding of data in the DCT domain

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
G06T9/00

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE
Etats contractants désignés lors du dépôt:

Bemerkungen:
Remarks:
Remarques:

The title of the application in Italian reads as follows:
"Architettura scalabile di calcolo parallelo della trasformata coseno discreta di blocchi di pixel di dimensioni diverse e compressione per codifica frattale"

VA/X00987/EP

Italian Text Pursuant to Art. 14.2

"ARCHITETTURA SCALABILE DI CALCOLO PARALLELO DELLA TRASFORMATA COSENO DISCRETA DI BLOCCHI DI PIXEL DI DIMENSIONI DIVERSE E COMPRESSIONE PER CODIFICA FRATTALE"

CAMPO DI APPLICAZIONE

L'invenzione concerne in generale i sistemi di elaborazione di immagini, da registrare e/o trasmettere e più in particolare a sistemi di compressione e codifica di immagine basati sul calcolo della trasformata coseno discreta (DCT) di blocchi di pixels di immagine.

L'invenzione è particolarmente utile in codificatori video secondo lo standard MPEG2 pur essendo applicabile anche ad altri sistemi.

BACKGROUND DELL'INVENZIONE

Il calcolo della trasformata coseno discreta (DCT) di una matrice di pixels di un'immagine rappresenta un passo fondamentale di elaborazione dei dati di immagine sui risultati della trasformazione DCT è quindi operata una divisione per una matrice detta di quantizzazione allo scopo di ridurre più o meno drasticamente l'ampiezza dei coefficienti DCT, come presupposto alla compressione dei dati che avviene in fase di codifica, secondo un certo protocollo di trasferimento dei dati video da trasmettere o memorizzare.

Il calcolo di trasformazione coseno discreta è tipicamente eseguito su blocchi o matrici di pixels in cui è suddivisa un'immagine.

Esigenze di velocità dei sistemi di elaborazione di immagini per la loro registrazione e/o trasmissione, impongono l'impiego di architetture hardware per velocizzare i vari passi del processo di elaborazione tra i quali, eminentemente, il passo di trasformazione coseno discreta per blocchi dei pixels di immagine.

L'implementazione hardware impone peraltro la predefinizione di alcuni parametri fondamentali, eminentemente della dimensione della matrice o blocco di pixels in cui suddividere un'immagine per le esigenze di elaborazione.

Tale predefinizione può rappresentare un pesante vincolo che limita le possibilità di ottimizzazione di un sistema di elaborazione, ad esempio di un codificatore MPEG2, o la sua adattabilità a condizioni di impiego molto diverse tra loro in termini di requisiti di prestazione.

È inoltre evidente l'enorme vantaggio sia in termini di riduzione dei costi che la disponibilità di un sistema integrato di elaborazione dei dati che possa essere programmato a calcolare in parallelo la DCT su più blocchi di pixels di dimensione selezionabile tra un certo numero di dimensioni disponibili.

SCOPI E SOMMARIO DELL'INVENZIONE

È evidente il bisogno e/o l'utilità di disporre di un metodo ed un dispositivo hardware di calcolo della trasformata coseno discreta (DCT) in parallelo su più blocchi di pixels, la cui architettura consenta una scalabilità della dimensione del blocco di pixels, ad esempio tale da consentire il calcolo della trasformata coseno discreta (DCT) per blocchi di 8x8 pixels, o di quattro blocchi di 4x4 pixels in parallelo, o di sedici blocchi di 2x2 pixels in parallelo, operando una selezione di dimensione del blocco *(size)*.

La scalabilità della dimensione del blocco di pixels e la possibilità di eseguire il calcolo della trasformata coseno discreta (DCT) in parallelo su più blocchi di dimensione frazionaria rispetto alla dimensione massima di blocco mediante una struttura hardware, è inoltre strumentale all'implementazione di schemi ed algoritmi di compressione di immagini "ibridi", ed altamente efficienti.

Ad esempio, in virtù della scalabilità della dimensione di blocco e della capacità di calcolo parallelo della DCT su più blocchi secondo la presente invenzione è implementabile su una codifica frattale applicata nel dominio della trasformata

coseno discreta (DCT) anziché nel dominio dello spazio dei dati di immagini, come è di consuetudine.

Rappresenta pertanto un aspetto importante dell'invenzione anche un nuovo metodo di compressione e codifica dei dati di un'immagine reso praticamente possibile dall'impiego di una struttura hardware di calcolo della trasformata coseno discreta (DCT) su blocchi di dimensione scalabile dell'invenzione e consistente essenzialmente nel

- suddividere un'immagine definendo due distinti tipi di blocchi di suddivisione: un primo tipo, di dimensione *N/i*N/i*, denominati blocchi *range* essenzialmente non sovrapponibili gli uni agli altri ed un secondo tipo, di dimensione *N*N*, denominati blocchi *domain* traslabili per intervalli di *N/i* pixels e sovrapponibili gli uni agli altri (cioè traslando sulla immagine originale la finestra che individua un blocco *domain* di un intervallo equivalente alla dimensione orizzontale e/o verticale di un blocco *range*);

- calcolare parallelamente la trasformata coseno discreta (DCT) dei 2^i blocchi *range* e di un relativo blocco *domain*;

- classificare i blocchi *range* trasformati in funzione della loro relativa complessità rappresentata dalla somma dei valori della terna di coefficienti AC;

applicare la trasformata frattale del dominio della DCT ai dati dei blocchi *range* con classificazione di complessità superiore ad una soglia prestabilita e memorizzare solo il coefficiente DC dei blocchi *range* con complessità inferiore a detta soglia, identificando un relativo blocco *domain* di appartenenza del blocco *range* in trasformazione tale da produrre la migliore approssimazione frattale dello stesso blocco *range* e di cui è calcolata la trasformata coseno discreta;

- calcolare un'immagine differenza tra ciascun blocco *range* e la sua approssimazione frattale;

- quantizzare detta immagine differenza nel dominio della DCT impiegando una tabella di quantizzazione predisposta in funzione delle caratteristiche della vista

umana;

- codificare l'immagine differenza quantizzata mediante un metodo di codifica basato sulla probabilità dei coefficienti di quantizzazione;

- memorizzare o trasmettere il codice di codifica per ciascun blocco *range* compresso nel dominio della DCT e il coefficiente DC di ciascun blocco *range* non compresso.

Indicando con $F_R(u,v)$, la DCT di un generico blocco *range*, nel dominio della DCT, di coefficienti AC indicano la complessità del blocco. Considerando il caso in cui $N=8$, i quattro coefficienti in alto a sinistra del blocco sono:

- il coefficiente DC che occupa la posizione 00;

- i tre coefficienti AC che occupano rispettivamente le posizioni 01, 10 e 11.

I coefficienti AC sono usati per decidere a quale classe appartenga un certo blocco *range*: se la somma dei loro valori assoluti è inferiore a una determinata soglia T, il blocco *range* in questione viene definito "a bassa attività"; se invece questa somma è uguale o maggiore di T, il blocco *range* viene definito "ad alta attività".

Per un blocco *range* a bassa attività, i coefficienti AC sono piccoli e pertanto possono essere omessi senza influenzare molto la fedeltà: in questo caso il blocco può essere approssimato memorizzandone soltanto il coefficiente DC.

Per un blocco *range* ad alta attività, il proseguimento di codifica frattale dell'invenzione consiste nel ricercare due trasformate lineari idonee (per esempio rotazioni, ribaltamenti, ecc.), se τ, ϕ, e un blocco *domain*, del quale la DCT è denotata con $F_D(u,v)$, che soddisfino, almeno approssimativamente, la seguente equazione:

$$F_R(u,v) = \tau \; o \; \phi \left(F_D(u,v)\right) \qquad (1)$$

Avendo identificato il blocco domain più simile o omologo al blocco *range* in elaborazione, se ne memorizzano i parametri $F_D(u,v), \tau, \phi$.

Si calcola quindi l'immagine differenza tra il blocco *range* e la sua posizione frattale

$$D(u,v) = F_R(u,v) - \tau \; o \; \phi \left(F_D(u,v)\right) \qquad (2)$$

Quantizzando l'immagine differenza D(u, v) si ottiene

$$D_Q(u,v) = INTEG\left[D(u,v) / Q(u,v)\right] \qquad (3)$$

dove:

- D_Q(u,v) è, nel dominio della DCT, l'immagine differenza quantizzata;
- Q(u, v) è una tabelle di quantizzazione progettata tenendo conto delle caratteristiche della vista umana;
- INTEG è una funzione che approssima il suo argomento all'intero più vicino.

Dopo la quantizzazione, la maggior parte dei coefficienti di D_Q(u, v) sono nulli. È facile progettare una codifica, ad esempio la codifica di Huffman, secondo le probabilità dei coefficienti. Alla fine si memorizza il codice da memorizzare o trasmettere. Quando ogni blocco *range* è stato codificato, la procedura di compressione è terminata.

L'oggetto dell'invenzione è più chiaramente definito nelle rivendicazioni annesse.

BREVE DESCRIZIONE DEI DISEGNI

I diversi aspetti implementativi dell'architettura scalabile del calcolo della trasformata coseno discreta dell'invenzione nonché di un efficiente metodo di compressione e codifica frattale da essa permesso, saranno più facilmente compresi attraverso la seguente descrizione dettagliata di una forma di

realizzazione dell'architettura dell'invenzione e delle diverse modalità di funzionamento secondo una selezione di dimensione *(size)* del blocco di pixels di suddivisione dell'immagine e facendo riferimento ai disegni allegati, nei quali:

la **Figura 1** è uno schema a blocchi di un codificatore attuante una compressione "ibrida" basata su codifica frattale e DCT, secondo la presente invenzione;

la **Figura 2** è il grafo di flusso di calcolo parallelo della DCT di sedici blocchi di dimensione 2x2;

la **Figura 3** illustra l'architettura per il calcolo di sedici DCT 2x2;

la **Figura 4** mostra l'ordinamento dei dati in ingresso per il calcolo di sedici DCT 2x2;

la **Figura 5** mostra la fase PROCESS del procedimento di calcolo di sedici DCT 2x2;

la **Figura 6** illustra l'architettura per il calcolo di quattro DCT 4x4;

la **Figura 7** mostra l'ordinamento dei dati in ingresso per il calcolo di quattro DCT 4x4;

la **Figura 8** mostra la fase PROCESS del procedimento di calcolo di quattro DCT 4x4;

la **Figura 9** illustra l'architettura per il calcolo di una DCT 8x8;

la **Figura 10** mostra l'ordinamento dei dati in ingresso per il calcolo di una DCT 8x8;

la **Figura 11** mostra la fase PROCESS del procedimento di calcolo di una DCT 8x8;

la **Figura 12** mostra l'architettura scalabile per il calcolo di una DCT 8x8 o di quattro DCT 4x4 in parallelo o di sedici DCT 2x2 in parallelo;

la **Figura 13** mostra la fase INPUT per l'architettura scalabile dell'invenzione;

la **Figura 14** mostra la fase PROCESS per l'architettura scalabile dell'invenzione;

la **Figura 15** mostra i blocchi che implementano la fase PROCESS per l'architettura scalabile dell'invenzione;

la **Figura 16** è uno schema di dettaglio del blocco QA;

la **Figura 17** è uno schema di dettaglio del blocco QB;

la **Figura 18** è uno schema di dettaglio del blocco QC;

la **Figura 19** è uno schema di dettaglio del blocco QD;

la **Figura 20** è uno schema di dettaglio del blocco QE;

la **Figura 21** è uno schema di dettaglio del blocco QF;

la **Figura 22** è uno schema di dettaglio del blocco QG;

la **Figura 23** illustra la fase ORDER per l'architettura scalabile dell'invenzione;

la **Figura 24** illustra la fase OUTPUT per l'architettura scalabile dell'invenzione.

Pur facendo riferimento in alcuni degli schemi illustrati nelle figure ad un'applicazione particolarmente significativa ed efficace dell'architettura di calcolo parallelo su blocchi di dimensione scalabile della DCT che comprende una fase di compressione per codifica frattale dei dati di immagine, resta inteso che il metodo e l'architettura di calcolo parallelo della trasformata coseno discreta (DCT) di una matrice bidimensionale di dati di ingresso per blocchi di dimensione scalabile, forniscono un'eccezionale libertà di implementazione di algoritmi di compressione particolarmente efficienti ed efficaci avvalendosi della scalabilità e della capacità di calcolo parallelo della DCT.

DESCRIZIONE DI ALCUNE FORME DI REALIZZAZIONE

Le fasi di partizionamento e di calcolo della trasformata coseno discreta di una matrice bidimensionale di dati di ingresso verrà ora descritta distintamente per ciascuna dimensione di blocco *range*, secondo un esempio di realizzazione, a partire dalla dimensione più piccola di 2x2 dei blocchi di dimensione frazionaria per i quali il calcolo della DCT è eseguito in parallelo, fino alla dimensione massima di 8x8 del blocco più grande.

A questa descrizione seguirà quindi la descrizione di un'architettura scalabile secondo necessità cambiando il valore della variabile globale *size*.

Il procedimento di calcolo parallelo di DCT dell'invenzione può essere scomposto in diverse fasi:

- Fase INPUT
- Fase PROCESS
- Fase ORDER
- Fase OUTPUT

che saranno descritte in successione per ciascun caso considerato.

L'operazione di trasformazione DCT può essere definita come segue.

Per una matrice dei dati in ingresso $x_{N*N} = [x_{i,j}], 0 \le i, j \le N-1$, la matrice di uscita $y_{N*N} = [y_{m,n}], 0 \le m, n \le N-1$, è definita da:

$$y_{m,n} = \frac{2}{N}\varepsilon(m)\varepsilon(n)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\cos\left(\frac{(2i+1)m}{2N}\pi\right)\cos\left(\frac{(2j+1)n}{2N}\pi\right), \qquad (4)$$

dove

$$\varepsilon(n) = \begin{cases} \frac{1}{\sqrt{2}} \; per \; n = 0 \\ 1 \; per \; 1 \le n \le N-1 \end{cases}$$

Per convenienza si assume che $N=2^i$, dove i e` un intero e $i \geq 1$. Si rimuovano $\varepsilon(m)$,$\varepsilon(n)$, e il valore di normalizzazione $2/N$ nell'Eq. (4), poiché è possibile reintrodurli in un passo successivo. Pertanto d'ora in poi si utilizzerà la seguente versione semplificata dell'Eq. (4):

$$y_{m,n} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} \cos\left[\frac{(2i+1)m}{2N}\pi\right] \cos\left[\frac{(2j+1)m}{2N}\pi\right], \qquad (5)$$

Calcolo parallelo di sedici DCT 2x2

Per $N = 2$ l'Eq.(5) diventa:

$$y_{m,n} = \sum_{i=0}^{1} \sum_{j=0}^{1} x_{i,j} \cos\left[\frac{(2i+1)m}{4}\pi\right] \cos\left[\frac{(2j+1)n}{4}\pi\right], \qquad (6)$$

Il grafo di flusso per una DCT 2x2 è mostrato in figura 2, nella quale $A=B=C=1$ e i dati in ingresso e in uscita sono i pixel nelle posizioni (0,0), (0,1), (1,0), (1,1).

Si consideri ora come calcolare in parallelo sedici DCT 2x2 in cui si suddivide un blocco 8x8.

Il procedimento è suddiviso in varie fasi, la cui veduta d'insieme è in figura 3.

In questa figura vengono evidenziate le trasformazioni eseguite sul blocchetto 2x2 costituito da (0,6), (0,7), (1,6), (1,7).

I pixel che costituiscono il **blocco in ingresso** vengono ordinati nella fase **INPUT** ed elaborati nella fase **PROCESS** in modo da ottenere i coefficienti delle sedici 2-D DCT su 4 campioni. Per esempio, la 2-D DCT del blocchetto (0,1) costituito da

{l[0],m[0],n[0],o[0]} è {a[0],b[0],c[0],d[0]}.

I coefficienti delle 2-D DCT vengono riordinati nella fase **ORDER** in otto vettori a otto componenti. Per esempio i coefficienti {a[0],b[0],c[0],d[0]} vanno a costituire il vettore l'.

I vettori così ottenuti vengono inviati alla fase **OUTPUT** per ottenere i coefficienti della

DCT 2x2, costituendo il **blocco di uscita**.

Fase INPUT

I pixel di ciascun blocchetto (i,j), con $0 \le i \le 1$ e $0 \le j \le 3$, vengono ordinati per costituire i vettori a 8 componenti ***l,m,n,o*** nel modo seguente:

- i pixel che occupano nel blocchetto la posizione (0,0) costituiscono il vettore ***l***;
- i pixel che occupano nel blocchetto la posizione (0,1) costituiscono il vettore ***m***;
- i pixel che occupano nel blocchetto la posizione (1,0) costituiscono il vettore ***n***;
- i pixel che occupano nel blocchetto la posizione (1,1) costituiscono il vettore ***o***;

Analogamente i pixel di ciascun blocchetto *(i,j)*, con $2 \le i \le 3$ e $0 \le j \le 3$, vengono ordinati per costituire i vettori a 8 componenti ***p,q,r,s*** nel modo seguente:

- i pixel che occupano nel blocchetto la posizione (0,0) costituiscono il vettore ***p***;
- i pixel che occupano nel blocchetto la posizione (0,1) costituiscono il vettore ***q***;
- i pixel che occupano nel blocchetto la posizione (1,0) costituiscono il vettore ***r***;
- i pixel che occupano nel blocchetto la posizione (1,1) costituiscono il vettore ***s***;

Questo ordinamento è mostrato in dettaglio in figura 4.

Si osservi, per esempio, che i pixel del blocchetto (0,3) vanno a costituire la componente 3 dei vettori ***l,m,n,o***.

Fase PROCESS

La fase **PROCESS** consiste nel calcolare in parallelo le sedici 2-D DCT elaborando i vettori a otto componenti ***l,m,...,s*** come mostrato in figura 5. Si osservi, per esempio, che i coefficienti della 2-D DCT del blocchetto (0,3) vanno a costituire la componente 3 dei vettori ***a,b,c,d*** di Figura 3.

Fase ORDER

La fase **ORDER** consiste nell'ordinamento

delle sequenze di uscita delle otto 2-D DCT in otto vettori ***l',m',...,s'*** così definiti:

$$q' = \begin{bmatrix} e[2] \\ f[2] \\ g[2] \\ h[2] \\ e[3] \\ f[3] \\ g[3] \\ h[3] \end{bmatrix}, \quad m' = \begin{bmatrix} a[2] \\ b[2] \\ c[2] \\ d[2] \\ a[3] \\ b[3] \\ c[3] \\ d[3] \end{bmatrix}, \quad n' = \begin{bmatrix} a[4] \\ b[4] \\ c[4] \\ d[4] \\ a[5] \\ b[5] \\ c[5] \\ d[5] \end{bmatrix}, \quad o' = \begin{bmatrix} a[6] \\ b[6] \\ c[6] \\ d[6] \\ a[7] \\ b[7] \\ c[7] \\ d[7] \end{bmatrix}$$

$$p' = \begin{bmatrix} e[0] \\ f[0] \\ g[0] \\ h[0] \\ e[1] \\ f[1] \\ g[1] \\ h[1] \end{bmatrix}, \quad q' = \begin{bmatrix} e[2] \\ f[2] \\ g[2] \\ h[2] \\ e[3] \\ f[3] \\ g[3] \\ h[3] \end{bmatrix}, \quad r' = \begin{bmatrix} e[4] \\ f[4] \\ g[4] \\ h[4] \\ e[5] \\ f[5] \\ g[5] \\ h[5] \end{bmatrix}, \quad s' = \begin{bmatrix} e[6] \\ f[6] \\ g[6] \\ h[6] \\ e[7] \\ f[7] \\ g[7] \\ h[7] \end{bmatrix}$$

Si osservi, per esempio, che i coefficienti della 2-D DCT del blocchetto (0,3) vanno a costituire le componenti 4,5,6,7 del vettore ***m'***.

Fase OUTPUT

Questa fase consiste nel riordinamento dei dati in uscita: a partire dai vettori di 8 componenti ***a,b,...,h*** si costruisce il vettore di 64 componenti

y così definito:

$$\begin{bmatrix} y[0] & y[1] & \cdots & y[7] \\ \vdots & \vdots & \vdots & \vdots \\ y[54] & y[55] & \cdots & y[63] \end{bmatrix} = \begin{bmatrix} l[0] & l[1] & l[4] & l[5] & m[0] & m[1] & m[4] & m[5] \\ l[2] & l[3] & l[6] & l[7] & m[2] & m[3] & m[6] & m[7] \\ n[0] & n[1] & n[4] & n[5] & o[0] & o[1] & o[4] & o[5] \\ n[2] & n[3] & n[6] & n[7] & o[2] & o[3] & o[6] & o[7] \\ p[0] & p[1] & p[4] & p[5] & q[0] & q[1] & q[4] & q[5] \\ p[2] & p[3] & p[6] & p[7] & q[2] & q[3] & q[6] & q[7] \\ r[0] & r[1] & r[4] & r[5] & s[0] & s[1] & s[4] & s[5] \\ r[2] & r[3] & r[6] & r[7] & s[2] & s[3] & s[6] & s[7] \end{bmatrix} \quad (7)$$

Calcolo parallelo di quattro DCT 4x4

Per $N = 4$ l' Eq. (5) diventa:

$$y_{m,n} = \sum_{i=0}^{3}\sum_{j=0}^{3} x_{i,j} \cos\left(\frac{(2i+1)m}{8}\pi\right)\cos\left(\frac{(2j+1)n}{8}\pi\right), \quad (8)$$

Ponendo:

$$Y_{16*1} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}, \; F_{16*1} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

dove:

$$y_i=[y_{i,0},y_{i,1},y_{i,2},y_{i,3}]'$$

$$\{f_{0,r}\}_{r=0}^{3} = DCT\left(\{A_{1,i}\}_{i=0}^{3}\right), \; \{f_{2,r}\}_{r=0}^{3} = DCT\left(\{B_{3,i}\}_{i=0}^{3}\right.$$

$$\{f_{1,r}\}_{r=0}^{3} = DCT\left(\{A_{3,i}\}_{i=0}^{3}\right),\ \{f_{3,r}\}_{r=0}^{3} = DCT\left(\{B_{1,i}\}_{i=0}^{3}\right)$$

$$\{A_{1,i}\}_{i=0}^{3} = \{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}\},$$

$$\{A_{3,i}\}_{i=0}^{3} = \{x_{1,0}, x_{3,1}, x_{0,2}, x_{2,3}\},$$

$$\{B_{3,i}\}_{i=0}^{3} = \{x_{2,0}, x_{0,1}, x_{3,2}, x_{1,3}\ ,$$

$$\{B_{1,i}\}_{i=0}^{3} = \{x_{3,0}, x_{2,1}, x_{1,2}, x_{0,3}$$

si può dimostrare che:

$$Y_{16*1} = (E_4)_{16} F_{16*1}, \tag{9}$$

dove

$$(E_4)_{16} = \begin{bmatrix} (H_0)_4 & (H_0)_4 & (H_0)_4 & (H_0)_4 \\ (H_1)_4 & (H_3)_4 & -(H_3)_4 & -(H_1)_4 \\ (H_2)_4 & -(H_2)_4 & -(H_2)_4 & (H_2)_4 \\ (H_3)_4 & -(H_1)_4 & (H_1)_4 & -(H_3)_4 \end{bmatrix}, \tag{10}$$

Le matrici $(H_i)_4$, i = 0,1,2,3 hanno le seguenti espressioni:

$$(H_0)_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, (H_1)_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix},$$

$$(H_2)_4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}, (H_3)_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0 \end{bmatrix}$$

La 1-D DCT è espressa dalla matrice (1-D DCT)$_4$=

$$C_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ C_8^1 & C_8^3 & -C_8^3 & -C_8^1 \\ C_4^1 & -C_4^1 & -C_4^1 & C_4^1 \\ C_8^3 & -C_8^1 & C_8^1 & -C_8^3 \end{bmatrix}, \text{ dove si è posto } C_n^m = \cos\left(\frac{m}{n}\pi\right)$$

Da queste equazioni si comprende che il calcolo di una DCT 4x4

Può essere suddiviso in due stadi:

- il calcolo di quattro 1-D DCT, ognuna eseguita su una sequenza opportuna di quattro pixel
- il calcolo della 2-D DCT a partire dalle quattro 1-D DCT

Questi due stadi vengono eseguiti in modo simile, in modo tale da essere realizzati da un unico hardware, che viene utilizzato due volte.

Si consideri ora come calcolare in parallelo quattro DCT 4x4. I 64 campioni complessivi sono ottenuti dai quattro blocchetti 4x4 in cui si suddivide un blocco 8x8.

Il procedimento èsuddiviso in varie fasi, a ciascuna della quali corrisponde un blocco architetturale. La veduta d'insieme è in figura 6. In questa figura vengono evidenziate le trasformazioni eseguite su ciascun blocchetto 4x4.

Fase INPUT

I pixel di ciascun quadrante $(i,j), 0 \le i, j \le 1$ vengono ordinati per costituire i

vettori:

$$A_1^{i,j} = \begin{bmatrix} x_{0,0}^{i,j} \\ x_{1,1}^{i,j} \\ x_{2,2}^{i,j} \\ x_{3,3}^{i,j} \end{bmatrix}, \quad A_3^{i,j} = \begin{bmatrix} x_{1,0}^{i,j} \\ x_{3,1}^{i,j} \\ x_{0,2}^{i,j} \\ x_{2,3}^{i,j} \end{bmatrix}, \quad B_3^{i,j} = \begin{bmatrix} x_{2,0}^{i,j} \\ x_{0,1}^{i,j} \\ x_{3,2}^{i,j} \\ x_{1,3}^{i,j} \end{bmatrix}, \quad B_1^{i,j} = \begin{bmatrix} x_{3,0}^{i,j} \\ x_{2,1}^{i,j} \\ x_{1,2}^{i,j} \\ x_{0,3}^{i,j} \end{bmatrix}$$

Dopo aver organizzato i dati in 16 vettori a 4 componenti, definiamo i vettori a 8 componenti ***l,m,n,o***, costituiti rispettivamente dalle componenti prima, seconda, terza, quarta degli otto vettori iniziali costituiti dai pixel dei quadranti 00 e 01 e i vettori ***p,q,r,s***, costituiti rispettivamente dalle componenti prima, seconda, terza, quarta degli otto vettori iniziali costituiti dai pixel dei quadranti 10 e 11. Precisamente:

$$l = \begin{bmatrix} A_1[0]^{0,0} \\ A_3[0]^{0,0} \\ B_3[0]^{0,0} \\ B_1[0]^{0,0} \\ A_1[0]^{0,1} \\ A_3[0]^{0,1} \\ B_3[0]^{0,1} \\ B_1[0]^{0,1} \end{bmatrix}, \quad m = \begin{bmatrix} A_1[1]^{0,0} \\ A_3[1]^{0,0} \\ B_3[1]^{0,0} \\ B_1[1]^{0,0} \\ A_1[1]^{0,1} \\ A_3[1]^{0,1} \\ B_3[1]^{0,1} \\ B_1[1]^{0,1} \end{bmatrix}, \quad n = \begin{bmatrix} A_1[2]^{0,0} \\ A_3[2]^{0,0} \\ B_3[2]^{0,0} \\ B_1[2]^{0,0} \\ A_1[2]^{0,1} \\ A_3[2]^{0,1} \\ B_3[2]^{0,1} \\ B_1[2]^{0,1} \end{bmatrix}, \quad o = \begin{bmatrix} A_1[3]^{0,0} \\ A_3[3]^{0,0} \\ B_3[3]^{0,0} \\ B_1[3]^{0,0} \\ A_1[3]^{0,1} \\ A_3[3]^{0,1} \\ B_3[3]^{0,1} \\ B_1[3]^{0,1} \end{bmatrix}$$

$$p = \begin{bmatrix} A_1[0]^{1,0} \\ A_3[0]^{1,0} \\ B_3[0]^{1,0} \\ B_1[0]^{1,0} \\ A_1[0]^{1,1} \\ A_3[0]^{1,1} \\ B_3[0]^{1,1} \\ B_1[0]^{1,1} \end{bmatrix}, \quad p = \begin{bmatrix} A_1[1]^{1,0} \\ A_3[1]^{1,0} \\ B_3[1]^{1,0} \\ B_1[1]^{1,0} \\ A_1[1]^{1,1} \\ A_3[1]^{1,1} \\ B_3[1]^{1,1} \\ B_1[1]^{1,1} \end{bmatrix}, \quad q = \begin{bmatrix} A_1[2]^{1,0} \\ A_3[2]^{1,0} \\ B_3[2]^{1,0} \\ B_1[2]^{1,0} \\ A_1[2]^{1,1} \\ A_3[2]^{1,1} \\ B_3[2]^{1,1} \\ B_1[2]^{1,1} \end{bmatrix}, \quad s = \begin{bmatrix} A_1[3]^{1,0} \\ A_3[3]^{1,0} \\ B_3[3]^{1,0} \\ B_1[3]^{1,0} \\ A_1[3]^{1,1} \\ A_3[3]^{1,1} \\ B_3[3]^{1,1} \\ B_1[3]^{1,1} \end{bmatrix}$$

Tenendo conto di come, a loro volta, sono stati definiti i vettori $A_1^{i,j}$, $A_3^{i,j}$,

$B_3^{i,j}, B_1^{i,j}$ si ottiene l'ordinamento mostrato in dettaglio in figura 7. Si osservi che in questa figura il blocco 8x8 originale è suddiviso nei quattro quadranti 4x4, nell'ambito di ciascun quadrante *(i,j)*, i pixel appartenenti ai vettori $A_1^{i,j}$, $A_3^{i,j}$, $B_3^{i,j}, B_1^{i,j}$ sono contrassegnati da colori diversi.

Secondo quanto descritto sopra, il calcolo di una DCT 4x4 può essere suddiviso in due stadi: di conseguenza, la fase **PROCESS**, che è l'unica fase in cui si svolgono operazioni aritmetiche, viene eseguita due volte:

- la prima volta, per calcolare in parallelo le sedici 1-D DCT;
- la seconda volta, per calcolare in parallelole quattro DCT 4x4 partendo dai coefficienti delle 1-D DCT.

Per poter "sapere" se è in corso il primo o il secondo stadio del calcolo, si utilizza la variabile *stage*.

Durante la fase **INPUT** la variabile *stage* viene aggiornata al valore 0.

All'ingresso della fase **PROCESS** vi sono 64 MUX controllati dalla variabile *stage*. Ciascun MUX riceve due ingressi:

- un pixel dell'immagine originale, proveniente dalla fase **INPUT** (questo ingresso viene selezionato quando *stage* = 0);
- un coefficiente di una 1-D DCT, proveniente dalla fase **ORDER** (questo ingresso viene selezionato quando *stage* = 1).

Fase PROCESS

Questa fase consiste nell'elaborare i vettori ***l***, ***m***, ..., ***s*** come mostrato in figura 8.

Nella figura 8 sono stati usati i seguenti simbolismi:

$$A = \begin{Bmatrix} 2C_8^1 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & per\ stage = 1 \end{Bmatrix}, \quad (11)$$

$$B = \begin{Bmatrix} 2C_8^3 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & per\ stage = 1 \end{Bmatrix}, \quad (12)$$

$$C = \begin{Bmatrix} 2C_4^1 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & per\ stage = 1 \end{Bmatrix}, \quad (13)$$

$$t = \begin{cases} 1 & per\ stage = 0 \\ 2 & per\ stage = 1 \end{cases}, \quad (14)$$

All'uscita della fase **PROCESS** vi sono 64 DEMUX controllati dalla variabile *stage*. I DEMUX hanno la funzione di indirizzare i dati secondo due possibilità:

- se *stage* = 0 il dato in ingresso a ciascun DEMUX è un coefficiente di una 1-D DCT; pertanto il dato deve essere processato ulteriormente e, a tal fine, viene inviato alla fase **ORDER**;
- se *stage* = 1 il dato in ingresso a ciascun DEMUX è un coefficiente di una 2-D DCT; pertanto il dato non deve essere processato ulteriormente e, a tal fine, viene inviato alla fase **OUTPUT**.

Fase ORDER

La fase **ORDER** consiste nell'ordinamento

delle sequenze di uscita delle otto 1-D DCT in otto vettori ***l'***, ***m'***,..., ***s'*** così definiti:

$$l' = \begin{bmatrix} f_0^{0,0} \\ f_0^{0,1} \end{bmatrix} = \begin{bmatrix} a[0] \\ b[0] \\ c[0] \\ d[0] \\ a[4] \\ b[4] \\ c[4] \\ d[4] \end{bmatrix}, \quad m' = \begin{bmatrix} f_1^{0,0} \\ f_1^{0,1} \end{bmatrix} = \begin{bmatrix} a[1] \\ b[1] \\ c[1] \\ d[1] \\ a[5] \\ b[5] \\ c[5] \\ d[5] \end{bmatrix},$$

$$n' = \begin{bmatrix} f_2^{0,0} \\ f_2^{0,1} \end{bmatrix} = \begin{bmatrix} a[2] \\ b[2] \\ c[2] \\ d[2] \\ a[6] \\ b[6] \\ c[6] \\ d[6] \end{bmatrix}, \quad o' = \begin{bmatrix} f_3^{0,0} \\ f_3^{0,1} \end{bmatrix} = \begin{bmatrix} a[3] \\ b[3] \\ c[3] \\ d[3] \\ a[7] \\ b[7] \\ c[7] \\ d[7] \end{bmatrix},$$

$$p' = \begin{bmatrix} f_0^{1,0} \\ f_0^{1,1} \end{bmatrix} = \begin{bmatrix} e[0] \\ f[0] \\ g[0] \\ h[0] \\ e[4] \\ f[4] \\ g[4] \\ h[4] \end{bmatrix}, \quad q' = \begin{bmatrix} f_1^{1,0} \\ f_1^{1,1} \end{bmatrix} = \begin{bmatrix} e[1] \\ f[1] \\ g[1] \\ h[1] \\ e[5] \\ f[5] \\ g[5] \\ h[5] \end{bmatrix},$$

$$r' = \begin{bmatrix} f_2^{1,0} \\ f_2^{1,1} \end{bmatrix} = \begin{bmatrix} e[2] \\ f[2] \\ g[2] \\ h[2] \\ e[6] \\ f[6] \\ g[6] \\ h[6] \end{bmatrix}, \quad s' = \begin{bmatrix} f_3^{1,0} \\ f_3^{1,1} \end{bmatrix} = \begin{bmatrix} e[3] \\ f[3] \\ g[3] \\ h[3] \\ e[7] \\ f[7] \\ g[7] \\ h[7] \end{bmatrix},$$

Dopo la fase **ORDER** la variabile *stage* viene aggiornata al valore 1. I dati in uscita dalla fase **ORDER** vengono inviati alla fase **PROCESS**.

Fase OUTPUT

Questa fase consiste nel riordinamento dei dati provenienti dalla seconda (cioè con *stage* =1) esecuzione della fase **PROCESS**: a partire da questi dati, che costituiscono i vettori di 8 componenti ***a***, ***b***,.., ***h***, si costruisce il blocco

di uscita Y_{N*N} così definito:

$$\begin{bmatrix} y[0] & y[1] & \cdots & y[7] \\ \vdots & \vdots & \vdots & \vdots \\ y[54] & y[55] & \cdots & y[63] \end{bmatrix} = \begin{bmatrix} a[0] & b[0] & c[0] & d[0] & e[0] & f[0] & g[0] & h[0] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a[3] & b[3] & c[3] & d[3] & e[3] & f[3] & g[3] & h[3] \\ e[4] & f[4] & g[4] & h[4] & a[4] & b[4] & c[4] & d[4] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e[7] & f[7] & g[7] & h[7] & a[7] & b[7] & c[7] & d[7] \end{bmatrix} \quad (15)$$

Le differenze principali fra l'hardware che realizza quattro DCT 4x4 e l'hardware che calcola sedici DCT 2x2 sono le seguenti:

- le sequenze in cui si devono ordinare i pixel di un blocco dell'immagine originale dipendono dalla dimensione della DCT richiesta;
- per eseguire sedici DCT 2x2 la fase **PROCESS** deve essere eseguita una sola volta; invece per eseguire quattro DCT 4x4 la fase **PROCESS** deve essere eseguita due volte;
- le operazioni eseguite nella fase PROCESS non sono sempre le stesse nei due casi.

Calcolo di una DCT 8x8

Per N = 8 l'Eq.(5) diventa:

$$y_{m,n} = \sum_{i=0}^{7}\sum_{j=0}^{7} x_{i,j} \cos\left[\frac{(2i+1)m}{16}\pi\right]\cos\left[\frac{(2j+1)n}{16}\pi\right], \qquad (16)$$

Ponendo:

$$Y_{64\times1} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_7 \end{bmatrix}, \; F_{64\times1} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_7 \end{bmatrix}$$

dove:

$$y_i = \begin{bmatrix} y_{i,0} & y_{i,1} & \cdots & y_{i,7} \end{bmatrix},$$

$$\{f_{0,r}\}_{r=0}^{7} = DCT\left(\{A_{1,i}\}_{i=0}^{7}\right), \; \{f_{4,r}\}_{r=0}^{7} = DCT\left(\{B_{7,i}\}_{i=0}^{7}\right),$$

$$\{f_{1,r}\}_{r=0}^{7} = DCT\left(\{A_{3,i}\}_{i=0}^{7}\right), \; \{f_{5,r}\}_{r=0}^{7} = DCT\left(\{B_{5,i}\}_{i=0}^{7}\right),$$

$$\{f_{2,r}\}_{r=0}^{7} = DCT\left(\{A_{5,i}\}_{i=0}^{7}\right), \; \{f_{6,r}\}_{r=0}^{7} = DCT\left(\{B_{3,i}\}_{i=0}^{7}\right),$$

$$\{f_{3,r}\}_{r=0}^{7} = DCT\left(\{A_{7,i}\}_{i=0}^{7}\right), \; \{f_{7,r}\}_{r=0}^{7} = DCT\left(\{B_{1,i}\}_{i=0}^{7}\right),$$

$$\{A_{1,i}\}_{i=0}^{7} = \{x_{0,0} \quad x_{1,1} \quad x_{2,2} \quad x_{3,3} \quad x_{4,4} \quad x_{5,5} \quad x_{6,6} \quad x_{7,7},$$

$$\{A_{3,i}\}_{i=0}^{7} = \{x_{1,0} \quad x_{4,1} \quad x_{7,2} \quad x_{5,3} \quad x_{2,4} \quad x_{0,5} \quad x_{3,6} \quad x_{6,7},$$

$$\{A_{5,i}\}_{i=0}^{7} = \{x_{2,0} \quad x_{7,1} \quad x_{3,2} \quad x_{1,3} \quad x_{6,4} \quad x_{4,5} \quad x_{0,6} \quad x_{5,7},$$

$$\{A_{7,i}\}_{i=0}^{7} = \{x_{3,0} \quad x_{5,1} \quad x_{1,2} \quad x_{7,3} \quad x_{0,4} \quad x_{6,5} \quad x_{2,6} \quad x_{4,7},$$

$$\{B_{7,i}\}_{i=0}^{7} = \{x_{4,0} \quad x_{2,1} \quad x_{6,2} \quad x_{0,3} \quad x_{7,4} \quad x_{1,5} \quad x_{5,6} \quad x_{3,7},$$

$$\{B_{5,i}\}_{i=0}^{7} = \{x_{5,0} \quad x_{0,1} \quad x_{4,2} \quad x_{6,3} \quad x_{1,4} \quad x_{3,5} \quad x_{7,6} \quad x_{2,7}\},$$

$$\{B_{3,i}\}_{i=0}^{7} = \{x_{6,0} \quad x_{3,1} \quad x_{0,2} \quad x_{2,3} \quad x_{5,4} \quad x_{7,5} \quad x_{4,6} \quad x_{1,7}\},$$

$$\{B_{1,i}\}_{i=0}^{7} = \{x_{7,0} \quad x_{6,1} \quad x_{5,2} \quad x_{4,3} \quad x_{3,4} \quad x_{2,5} \quad x_{1,6} \quad x_{0,7}\}$$

si può dimostrare che:

$$Y_{64\times 1} = (E_8)_{64} F_{64\times 1}, \tag{17}$$

dove

$$(E_8)_{64} = \begin{bmatrix} (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\ (H_1)_8 & (H_3)_8 & (H_5)_8 & (H_7)_8 & -(H_7)_8 & -(H_5)_8 & -(H_3)_8 & -(H_1)_8 \\ (H_2)_8 & (H_6)_8 & -(H_6)_8 & -(H_2)_8 & -(H_2)_8 & -(H_6)_8 & (H_6)_8 & (H_2)_8 \\ (H_3)_8 & -(H_7)_8 & -(H_1)_8 & -(H_5)_8 & (H_5)_8 & (H_1)_8 & (H_7)_8 & -(H_3)_8 \\ (H_4)_8 & -(H_4)_8 & -(H_4)_8 & (H_4)_8 & (H_4)_8 & -(H_4)_8 & -(H_4)_8 & (H_4)_8 \\ (H_5)_8 & -(H_1)_8 & (H_7)_8 & (H_3)_8 & -(H_3)_8 & -(H_7)_8 & (H_1)_8 & -(H_5)_8 \\ (H_6)_8 & -(H_2)_8 & (H_2)_8 & -(H_6)_8 & -(H_6)_8 & (H_2)_8 & -(H_2)_8 & (H_6)_8 \\ (H_7)_8 & -(H_5)_8 & (H_3)_8 & -(H_1)_8 & (H_1)_8 & -(H_3)_8 & (H_5)_8 & -(H_7)_8 \end{bmatrix} \tag{18}$$

Le matrici $(H_i)_s$, i=0,1,..,7 hanno le seguenti espressioni:

$$(H_0)_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$(H_1)_8 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix},$$

$$(H_2)_8 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix},$$

$$(H_3)_8 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 \end{bmatrix},$$

$$(H_4)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \end{bmatrix},$$

$$(H_5)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 \end{bmatrix},$$

$$(H_6)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$(H_7)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La 1-D DCT è espressa dalla matrice

$$(1-D\ DCT)_8 = C_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ C_{16}^1 & C_{16}^3 & C_{16}^5 & C_{16}^7 & -C_{16}^7 & -C_{16}^5 & -C_{16}^3 & -C_{16}^1 \\ C_8^3 & C_8^3 & C_8^5 & C_8^7 & C_8^7 & C_8^5 & C_8^3 & C_8^1 \\ C_{16}^3 & -C_{16}^7 & -C_{16}^1 & -C_{16}^5 & C_{16}^5 & C_{16}^1 & C_{16}^7 & -C_{16}^3 \\ C_4^1 & -C_4^1 & -C_4^1 & C_4^1 & C_4^1 & -C_4^1 & -C_4^1 & C_4^1 \\ C_{16}^5 & -C_{16}^1 & C_{16}^7 & C_{16}^3 & -C_{16}^3 & -C_{16}^7 & C_{16}^1 & -C_{16}^5 \\ C_8^3 & -C_8^1 & C_8^1 & C_8^5 & C_8^5 & C_8^1 & C_8^1 & C_8^3 \\ C_{16}^7 & -C_{16}^5 & C_{16}^3 & -C_{16}^1 & C_{16}^1 & -C_{16}^3 & -C_{16}^5 & -C_{16}^7 \end{bmatrix}$$

dove si è posto:

$$C_n^m = \cos(\frac{m}{n}\pi)$$

Da queste equazioni si comprende che il calcolo di una DCT 8x8 può essere suddiviso in due stadi:

- il calcolo di otto 1-D DCT, ognuna eseguita su una sequenza opportuna di otto pixel
- il calcolo della 2-D DCT a partire dalle otto 1-D DCT

Questi due stadi possono essere eseguiti in modo simile, in modo tale da essere realizzati da un unico hardware, che viene utilizzato due volte.

Il procedimento è suddiviso in varie fasi, a ciascuna della quali corrisponde un blocco architetturale. La veduta d'insieme è in figura 9.

Fase INPUT

I pixel del blocco 8x8 in ingresso vengono ordinati per costituire i vettori a 8 componenti **l**, ***m***, ***n***, ***o***, ***p***, ***q***, ***r***, ***s***:

$$l = \begin{bmatrix} A_1[0] \\ A_3[0] \\ A_5[0] \\ A_7[0] \\ B_7[0] \\ B_5[0] \\ B_3[0] \\ B_1[0] \end{bmatrix}, \; m = \begin{bmatrix} A_1[1] \\ A_3[1] \\ A_5[1] \\ A_7[1] \\ B_7[1] \\ B_5[1] \\ B_3[1] \\ B_1[1] \end{bmatrix}, \ldots, s = \begin{bmatrix} A_1[7] \\ A_3[7] \\ A_5[7] \\ A_7[7] \\ B_7[7] \\ B_5[7] \\ B_3[7] \\ B_1[7] \end{bmatrix}$$

Tenendo conto di come, a loro volta, sono stati definiti i vettori A_1, A_3, A_5, A_7, B_7, B_5, B_3, B_1 si ottiene l'ordinamento mostrato in dettaglio in figura 10. Si osservi che in questa figura i pixel appartenenti ai vettori A_1, A_3, A_5, A_7, B_7, B_5, B_3, B_1 sono contrassegnati da colori diversi.

Come mostrato sopra, il calcolo di una DCT 8x8 può essere suddiviso in due stadi: di conseguenza, la fase **PROCESS**, che è l'unica fase in cui si svolgono operazioni aritmetiche, viene eseguita due volte:

- la prima volta, per calcolare in parallelo le sedici 1-D DCT;
- la seconda volta, per calcolare la DCT 8x8 partendo dai coefficienti delle 1-D DCT.

Per poter "sapere" se è in corso il primo o il secondo stadio del calcolo, si utilizza la variabile *stage*.

Durante la fase **INPUT** la variabile *stage* viene aggiornata al valore 0.

All'ingresso della fase **PROCESS** vi sono 64 MUX controllati dalla variabile *stage*. Ciascun MUX riceve due ingressi:

- un pixel dell'immagine originale, proveniente dalla fase **INPUT** (questo ingresso viene selezionato quando *stage* = 0);
- un coefficiente di una 1-D DCT, proveniente dalla fase **ORDER** (questo ingresso viene selezionato quando *stage* = 1).

Fase PROCESS

Questa fase consiste nell'elaborare i vettori ***l***, ***m***, ..., ***s*** come mostrato in figura 11.

Nella figura 11 sono stati usati i seguenti simbolismi:

$$A = \begin{cases} 2C_8^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_2)_8 & per\ stage = 1 \end{cases} \quad (19)$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times 8} & per\ stage = 0 \\ -(H_6)_8 & per\ stage = 1 \end{cases} \quad (20)$$

$$C = \begin{cases} 2C_4^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_4)_8 & per\ stage = 1 \end{cases} \quad (21)$$

$$t = \begin{cases} 1 & per\ stage = 0 \\ 2 & per\ stage = 1 \end{cases} \quad (22)$$

All' uscita della fase **PROCESS** vi sono 64 DEMUX controllati dalla variabile *stage*. I DEMUX hanno la funzione di indirizzare i dati secondo due possibilità:

- se *stage* = 0 il dato in ingresso a ciascun DEMUX è un coefficiente di una 1-D DCT; pertanto il dato deve essere processato ulteriormente e, a tal fine, viene inviato alla fase **ORDER**;

- se *stage* = 1 il dato in ingresso a ciascun DEMUX è un coefficiente della 2-D DCT; pertanto il dato non deve essere processato ulteriormente e, a tal fine, viene inviato alla fase **OUTPUT**.

Fase ORDER

La fase **ORDER** consiste nell'ordinamento delle sequenze di uscita delle otto 1-D DCT in otto vettori ***l'***, ***m'***, ..., ***s'*** così definiti:

$$l' = f_0 = \begin{bmatrix} a[0] \\ b[0] \\ c[0] \\ d[0] \\ e[0] \\ f[0] \\ g[0] \\ h[0] \end{bmatrix}, \; m' = f_1 = \begin{bmatrix} a[1] \\ b[1] \\ c[1] \\ d[1] \\ e[1] \\ f[1] \\ g[1] \\ h[1] \end{bmatrix}, \ldots\ldots, \; q' = f_7 = \begin{bmatrix} a[7] \\ b[7] \\ c[7] \\ d[7] \\ e[7] \\ f[7] \\ g[7] \\ h[7] \end{bmatrix}$$

Dopo la fase **ORDER** la variabile *stage* viene aggiornata al valore 1. I dati in uscita dalla fase **ORDER** vengono inviati alla fase **PROCESS**.

Fase OUTPUT

Questa fase consiste nel riordinamento dei dati provenienti dalla seconda (cioè con *stage* =1) esecuzione della fase **PROCESS**: a partire da questi dati, che costituiscono i vettori di 8 componenti ***a***, ***b***, ..., ***h***, si costruisce il blocco di uscita Y_{N*N} così definito:

$$\begin{bmatrix} y[0] & y[1] & \cdots & y[7] \\ \vdots & \vdots & \vdots & \vdots \\ y[54] & y[55] & \vdots & y[63] \end{bmatrix} = \begin{bmatrix} a[0] & b[0] & c[0] & d[0] & e[0] & f[0] & g[0] & h[0] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a[7] & b[7] & c[7] & d[7] & e[7] & f[7] & g[7] & h[7] \end{bmatrix}, \quad (23)$$

Le differenze principali fra l'hardware che realizza una DCT 8x8 e l'hardware che calcola quattro DCT 4x4 sono le seguenti:

- le sequenze in cui si devono ordinare i pixel di un blocco dell'immagine originale dipendono dalla dimensione della DCT richiesta;
- le operazioni eseguite nella fase **PROCESS** non sono sempre le stesse nei due casi.

Procedimento di calcolo della DCT per blocchi di dimensione scalabile (DCT 8x8, DCT 4x4 e DCT 2x2)

Dai procedimenti già esposti si ricava un algoritmo per calcolare, a scelta, una DCT 8x8 oppure

quattro DCT 4x4 (in parallelo) oppure sedici DCT 2x2 (in parallelo).

Questa scelta viene indicata dall'utente assegnando il valore della variabile globale *"size"*:

$$\text{size} = \begin{cases} 0 \text{ per una DCT 8x8} \\ 1 \text{ per quattro DCT 4x4} \\ 2 \text{ per sedici DCT 2x2} \end{cases}$$

Il procedimento è suddiviso in varie fasi (indipendentemente dal valore di *"size"*), a ciascuna della quali corrisponde un blocco architetturale. La veduta d'insieme è in figura 12.

Ogni fase è stata riorganizzata in modo tale da fornire i risultati parziali corrispondenti al valore di prescelto, minimizzando le ridondanze. A volte le operazioni da eseguire sono diverse a seconda del valore di size: in questi casi l'architettura prevede un MUX il cui ingresso di controllo è size. Si esaminino ora le varie fasi mettendo in evidenza le differenze rispetto alle architetture già descritte:

Fase INPUT

La finalità di questa fase, descritta in Figura 13, è di ordinare i dati in modo opportuno per poter calcolare, partendo da loro, le 1-D DCT. Questo viene fatto

ricevendo in ingresso i valori di luminanza dei pixel (matrice x8x8), e ordinandoli negli otto vettori a otto componenti ***l, m, ..., s***.

Per esempio:

$$l[0] = x_{0,0}$$

$$l[1] = \begin{cases} x_{1,2} & per\ size = 0 o 1 \\ x_{2,0} & per\ size = 2 \end{cases}$$

$$s[7] = \begin{cases} x_{0,7} & per\ size = 0 \\ x_{4,7} & per\ size = 1 \\ x_{7,7} & per\ size = 2 \end{cases}$$

Fase PROCESS con *stage* = 0

Questa fase consiste nel calcolare in parallelo le otto 1-D DCT elaborando i vettori ***l, m, ..., s*** come mostrato in figura 14.

In questa figura si può notare l'uso di 16 MUX controllati da size.

Gli otto MUX sulla sinistra servono a poter "bypassare" le operazioni necessarie solo per il calcolo della DCT 8x8; pertanto il "bypass" avviene per *size* = 1 o 2, mentre non avviene per size = 0.

Gli otto MUX sulla destra servono a inviare in uscita solo il risultato corrispondente al valore di size prescelto.

$$t = \begin{cases} 1 & per\ stage = 0 \\ 2 & per\ stage = 1 \end{cases}, \qquad (24)$$

$$A=\begin{cases} 2C_8^1 \times I_{8\times 8} & per\ stage, size = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ stage, size = (0,2)\ oppure\ (1,2) \\ (H_2)_8 & per\ stage, size = (1,0) \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & per\ stage, size = (1,1) \end{cases}, \quad (25)$$

$$B=\begin{cases} 2C_8^3 \times I_{8\times 8} & per\ stage, size = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ stage, size = (0,2)\ oppure\ (1,2) \\ -(H_6)_8 & per\ stage, size = (1,0) \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & per\ stage, size = (1,1) \end{cases}, \quad (26)$$

$$C=\begin{cases} 2C_4^1 \times I_{8\times 8} & per\ stage, size = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ stage, size = (0,2)\ oppure\ (1,2) \\ (H_4)_8 & per\ stage, size = (1,0) \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & per\ stage, size = (1,1) \end{cases}, \quad (27)$$

$$D=\begin{cases} 2C_{16}^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_1)_8 & per\ stage = 1 \end{cases} \quad (28)$$

$$E=\begin{cases} 2C_8^3 \times I_{8\times 8} & per\ stage = 0 \\ (H_5)_8 & per\ stage = 1 \end{cases} \quad (29)$$

$$F=\begin{cases} 2C_{16}^7 \times I_{8\times 8} & per\ stage = 0 \\ -(H_7)_8 & per\ stage = 1 \end{cases} \quad (30)$$

$$G=\begin{cases} 2C_{16}^5 \times I_{8\times 8} & per\ stage = 0 \\ -(H_3)_8 & per\ stage = 1 \end{cases} \quad (31)$$

Lo schema in figura 14 può essere scomposto nei blocchi mostrati in figura 15.

Il blocco QA, per esempio, riceve in ingresso due vettori a otto componenti (ogni componente è un pixel, che può aver già subito elaborazioni) e invia come uscite due vettori a otto componenti: il primo vettore è la somma dei due vettori in ingresso, mentre il secondo vettore è la differenza fra i due vettori in ingresso successivamente elaborata tramite l'operatore lineare A.

Si noti che gli operatori A,B,C,D,E,F,G sono matrici 8x8.

Passando a un livello inferiore di generalizzazione i blocchi QA, QB, QC sono mostrati in dettaglio rispettivamente nelle figure 16, 17, 18. In queste figure i MUX sono controllati da tre bit, che corrispondono alla variabile stage (che può valere 0 o 1, e pertanto è rappresentata da un bit) e alla variabile size (che può valere 0,1 o 2, e pertanto è rappresentata da due bit).

I blocchi QD, QE, QF, QG sono mostrati in dettaglio rispettivamente nelle figure 19, 20, 21, 22. In queste figure i MUX sono controllati da un bit, che corrisponde alla variabile *stage*.

Fase ORDER

La fase ORDER, descritta in Figura 23, consiste nell'ordinamento delle sequenze di uscita delle otto 1-D DCT in otto vettori ***l', m', ..., s'***. Per esempio:

$$l'[0] = a[0];$$
$$l'[1] = b[0];$$

$$\vdots$$

$$l'[4] = \begin{cases} e[0] & per\ size = 0 \\ a[4] & per\ size = 1 \\ a[1] & per\ size = 2 \end{cases}$$

$$\vdots$$

$$s'[7] = h[7];$$

Fase OUTPUT

Questa fase, descritta in Figura 24, consiste nel riordinamento dei dati provenienti dalla seconda (cioè con stage = 1) esecuzione della fase PROCESS: a partire da questi dati, che costituiscono i vettori di 8 componenti ***a, b, ... , h*** si costruisce il blocco di uscita yNxN.

Per esempio:

$$y[0] = \begin{cases} a[0] & per\ size = 0o1 \\ l[2] & per\ size = 2 \end{cases}$$

$$y[1] = \begin{cases} b[0] & per\ size = 0o1 \\ l[1] & per\ size = 2 \end{cases}$$

$$\vdots$$

$$y[63] = \begin{cases} h[7] & per\ size = 0 \\ d[7] & per\ size = 1 \\ s[7] & per\ size = 2 \end{cases}$$

DESCRIZIONE DELLE FIGURE

Uno schema funzionale a blocchi di un compressore-codificatore di immagini secondo la presente invenzione può essere rappresentato come mostrato in *Figura 1:* Esso esegue pertanto una compressione "ibrida" che si basa essenzialmente su una codifica frattale eseguita nel dominio della DCT. Ciò è reso possibile dalla peculiare architettura di calcolo parallelo della DCT su blocchi di pixel di dimensione scalabile, come descritto sopra.

Vengono qui di seguito descritte singolarmente le altre figure.

Figura 2: Grafo di flusso della DCT 2x2

Questo è il blocco ``basè', che compare ripetutamente nella fase PROCESS di tutte le DCT NxN, dove è una potenza di 2.

In particolare:

* Il grafo di flusso per una DCT 2x2 è mostrato in figura 2, nella quale A = B = C = 1 e i dati in ingresso e in uscita sono i pixel nelle posizioni (0,0), (0,1), (1,0), (1,1).

* per sedici DCT 2x2 gli ingressi e le uscite sono vettori a 8 componenti e sono usati i seguenti simbolismi e si pone A = B = C = I8x8

* per quattro DCT 4x4 gli ingressi e le uscite sono vettori a 8 componenti e sono usati i seguenti simbolismi:

$$A = \begin{cases} 2C_8^1 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & per\ stage = 1 \end{cases}, \tag{32}$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & per\ stage = 1 \end{cases}, \tag{33}$$

$$C = \begin{cases} 2C_4^1 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & per\ stage = 1 \end{cases}, \tag{34}$$

* per una DCT 8x8 gli ingressi e le uscite sono vettori a 8 componenti e sono usati i seguenti simbolismi:

$$A = \begin{cases} 2C_8^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_2)_8 & per\ stage = 1 \end{cases}, \tag{35}$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times8} & per\ stage = 0 \\ -(H_6)_8 & per\ stage = 1 \end{cases}, \quad (36)$$

$$C = \begin{cases} 2C_4^1 \times I_{8\times8} & per\ stage = 0 \\ (H_4)_8 & per\ stage = 1 \end{cases}, \quad (37)$$

* Nell'architettura scalabile per il calcolo di una DCT 8x8 oppure quattro DCT 4x4 (in parallelo) oppure sedici DCT 2x2 (in parallelo) gli ingressi e le uscite sono vettori a 8 componenti e sono usati i seguenti simbolismi:

$$A = \begin{cases} 2C_8^1 \times I_{8\times8} & per\ (stage, size) = (0,0)\ oppure\ (0,1) \\ I_{8\times8} & per\ (stage, size) = (0,2)\ oppure\ (1,2) \\ (H_2)_8 & per\ (stage, size)\ (1,0) \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & per\ (stage, size)\ (1,1) \end{cases}, \quad (38)$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times8} & per\ (stage, size) = (0,0)\ oppure\ (0,1) \\ I_{8\times8} & per\ (stage, size) = (0,2)\ oppure\ (1,2) \\ -(H_6)_8 & per\ (stage, size)\ (1,0) \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & per\ (stage, size)\ (1,1) \end{cases}, \quad (39)$$

$$C = \begin{cases} 2C_4^1 \times I_{8\times 8} & per\ (stage, size) = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ (stage, size) = (0,2)\ oppure\ (1,2) \\ (H_4)_8 & per\ (stage, size)\ (1,0) \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & per\ (stage, size)\ (1,1) \end{cases}, \quad (40)$$

Figura 3: Architettura per il calcolo di sedici DCT 2x2

I pixel che costituiscono il blocco in ingresso vengono ordinati nella fase INPUT ed elaborati nella fase PROCESS in modo da ottenere i coefficienti delle sedici 2-D DCT su 4 campioni. Per esempio, la 2-D DCT del blocchetto (0,1) costituito da

{l[0],m[0],n[0],o[0]} è {a[0],b[0],c[0],d[0]}.

I coefficienti delle 2-D DCT vengono riordinati nella fase ORDER in otto vettori a otto componenti. Per esempio i coefficienti {a[0],b[0],c[0],d[0]} vanno a costituire il vettore *l'*.

I sedici vettori a due componenti così ottenuti vengono inviati alla fase PROCESS per ottenere i coefficienti della DCT 2x2. Questi coefficienti, riordinati nella fase OUTPUT, costituiscono il blocco di uscita.

Figura 4: Ordinamento dei dati in ingresso per il calcolo di sedici DCT 2x2

Questa figura mostra come i pixel del blocco 8x8 in ingresso vengono ordinati per costituire i vettori a 8 componenti l, m, ... s. In ciascun quadrante *(i,j)*, con $0 \leq i,j \leq 3$, i pixel appartenenti ai vettori sono contrassegnati da colori diversi. Per esempio:

$$\left\{A_{i,k}^{0,0}\right\}_{k=0}^{1} = \left\{x_{0,0}, x_{1,1}\right\}$$

Da ciascuno di questi vettori, le componenti con lo stesso indice (ossia i pixel con

lo stesso indice di colonna) vanno a formare uno stesso vettore a quattro componenti. Ad esempio il vettore l è costituito dagli elementi {A1[0], B1[0]}.

Ne risulta che ciascun pixel del blocco 8x8 in ingresso viene a costituire una componente di uno dei vettori l, m, n, o, p, q, r, s.

Figura 5: Fase PROCESS per il calcolo di sedici DCT 2x2

Questa fase consiste nell'elaborare i vettori a 8 componenti l, m, .. , s. La fase PROCESS}, che è l'unica fase in cui si svolgono operazioni aritmetiche, viene eseguita una sola volta, per calcolare in parallelo le sedici 2-D DCT.

Figura 6: Architettura per il calcolo di quattro DCT 4x4

I pixel che costituiscono il blocco in ingresso vengono ordinati nella fase INPUT ed elaborati nella fase PROCESS in modo da ottenere i coefficienti delle sedici 1-D DCT su 4 campioni. Per esempio, la 1-D DCT della sequenza *{l[0],m[0],n[0],o[0]} è{a[0],b[0],c[0],d[0]}*.

I coefficienti delle 1-D DCT vengono riordinati nella fase ORDER in otto vettori a otto componenti. Per esempio i coefficienti {a[0],b[0],c[0],d[0]} vanno a costituire il vettore l'.

I quattro vettori a quattro componenti così ottenuti vengono inviati alla fase PROCESS per ottenere i coefficienti della DCT 4x4. Questi coefficienti, riordinati nella fase OUTPUT}, costituiscono il blocco di uscita.

Figura 7: Ordinamento dei dati in ingresso per il calcolo di quattro DCT 4x4

Questa figura mostra come i pixel del blocco 8x8 in ingresso vengono ordinati per costituire i vettori a 8 componenti l, m, ..., s.

In ciascun quadrante (i,j), con $0 \leq i,j \leq 3$, i pixel appartenenti ai vettori

sono contrassegnati da colori diversi. Per esempio:

$$\left\{A_{r,k}^{0,0}\right\}_{k=0}^{3} = \left\{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}\right\}$$

Da ciascuno di questi vettori, le componenti con lo stesso indice (ossia i pixel con lo stesso indice di colonna) vanno a formare uno stesso vettore a quattro componenti. Ad esempio il vettore 1 è costituito dagli elementi *{A1[0], A3[0], B3[0], B1[0]}*.

Ne risulta che ciascun pixel del blocco 8x8 in ingresso viene a costituire una componente di uno dei vettori l, m, n, o, p, q, r, s.

Figura 8: Fase PROCESS per il calcolo di quattro DCT 4x4

Questa fase consiste nell'elaborare i vettori a 8 componenti *l, m, ... s*.

La fase PROCESS, che è l'unica fase in cui si svolgono operazioni aritmetiche, viene eseguita due volte:

* la prima volta (stage = 0), per calcolare in parallelo le sedici 1-D DCT;

* la seconda volta (stage = 1), per calcolare la DCT 8x8 partendo dai coefficienti delle 1-D DCT.

$$A = \begin{cases} 2C_8^1 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & per\ stage = 1 \end{cases}, \qquad (41)$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & per\ stage = 1 \end{cases}, \qquad (42)$$

$$C = \begin{cases} 2C_4^1 \times I_{8\times 8} & per\ stage = 0 \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & per\ stage = 1 \end{cases}, \qquad (43)$$

$$t = \begin{cases} 1 & per\ stage = 0 \\ 2 & per\ stage = 1 \end{cases}, \qquad (44)$$

Figura 9: Architettura per il calcolo di una DCT 8x8

I pixel che costituiscono il blocco in ingresso vengono ordinati nella fase INPUT ed elaborati nella fase PROCESS in modo da ottenere i coefficienti delle otto 1-D DCT su 8 campioni. Per esempio, la 1-D DCT della sequenza {l[0],m[0], ... ,s[0]} è {a[0],b[0], ... ,h[0]}.

I coefficienti delle 1-D DCT vengono riordinati nella fase ORDER in otto vettori a otto componenti. Per esempio i coefficienti {a[0],b[0], ... ,h[7]} vanno a costituire il vettore l'.

Gli otto vettori a otto componenti così ottenuti vengono inviati alla fase PROCESS per ottenere i coefficienti della DCT 8x8. Questi coefficienti, riordinati nella fase OUTPUT, costituiscono il blocco di uscita.

Figura 10: Ordinamento dei dati in ingresso per il calcolo di una DCT 8x8

Questa figura mostra come i pixel del blocco 8x8 in ingresso vengono ordinati per costituire i vettori a 8 componenti l, m, ... , s. I pixel appartenenti ai vettori *A1, A3, A5, A7, B7, B5, B3, B1* sono contrassegnati da colori diversi. Per esempio:

$$\{A_{i,j}\}_{i=0}^{7} = \{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}, x_{4,4}, x_{5,5}, x_{6,6}, x_{7,7}\}$$

Da ciascuno di questi vettori, le componenti con lo stesso indice (ossia i pixel con lo stesso indice di colonna) vanno a formare uno stesso vettore a otto componenti. Ad esempio il vettore l è costituito dagli elementi { A1[0], A3[0], ... , B1[0]}.

Ne risulta che ciascun pixel del blocco 8x8 in ingresso viene a costituire una componente di uno dei vettori l, m, n, o, p, q, r, s.

Figura 11: Fase PROCESS per il calcolo di una DCT 8x8

Questa fase consiste nell'elaborare i vettori a 8 componenti l, m, ..., s.

La fase PROCESS, che è l'unica fase in cui si svolgono operazioni aritmetiche, viene eseguita due volte:

* la prima volta (stage = 0), per calcolare in parallelo le sedici 1-D DCT;

* la seconda volta (stage = 1), per calcolare la DCT 8x8 partendo dai coefficienti delle 1-D DCT.

Nella figura 11 sono stati usati i seguenti simbolismi:

$$A = \begin{cases} 2C_8^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_2)_8 & per\ stage = 1 \end{cases}, \tag{45}$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times 8} & per\ stage = 0 \\ -(H_6)_8 & per\ stage = 1 \end{cases}, \tag{46}$$

$$C = \begin{cases} 2C_4^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_4)_8 & per\ stage = 1 \end{cases}, \tag{47}$$

$$t = \begin{cases} 1 & per\ stage = 0 \\ 2 & per\ stage = 1 \end{cases}, \tag{48}$$

Figura 12: Architettura scalabile per il calcolo di una DCT 8x8 o di quattro DCT 4x4 o di sedici DCT 2x2

I pixel che costituiscono il blocco in ingresso vengono ordinati nella fase INPUT ed elaborati nella fase PROCESS, dove vengono calcolate:

* le DCT 1-D (per stage = 0, cioè per la DCT 8x8, e per stage = 1, cioè per le DCT

4x4)

* direttamente le DCT 2-D (per stage} = 2, cioè per le DCT 2x2)

Nei casi stage = 0 e stage = 1 i coefficienti vengono poi riordinati nella fase ORDER in otto vettori a otto componenti, che vengono inviati alla fase PROCESS per ottenere i coefficienti della DCT 2-D. Questi coefficienti, riordinati nella fase OUTPUT, costituiscono il blocco di uscita.

Nel caso stage = 2 i coefficienti vengono trasmessi direttamente alla fase OUTPUT, dove vengono riordinati per costituire il blocco di uscita.

Figura 13: Fase INPUT per l'architettura scalabile

Gli ingressi sono i 64 pixel che costituiscono il blocco in ingresso.

L'ordinamento degli ingressi viene operato per mezzo di MUX controllati dalla variabile size.

Le 64 uscite sono le componenti degli otto vettori a otto componenti ***l, m, ... , s***.

Figura 14: Fase PROCESS per l'architettura scalabile.

Questa fase consiste nel calcolare in parallelo le otto 1-D DCT elaborando i vettori ***l, m, ... s*** come mostrato in figura 11.

In questa figura si può notare l'uso di 16 MUX controllati da size.

Gli otto MUX sulla sinistra servono a poter ``bypassarè' le operazioni necessarie solo per il calcolo della DCT 8x8; pertanto il ``bypass» avviene per stage = 1 o 2, mentre non avviene per stage = 0.

Gli otto MUX sulla destra servono a inviare in uscita solo il risultato corrispondente al valore di size prescelto.

Nella Figura 14 sono stati usati i seguenti simbolismi:

$$t = \begin{cases} 1 & per\ stage = 0 \\ 2 & per\ stage = 1 \end{cases}, \tag{49}$$

$$A = \begin{cases} 2C_8^1 \times I_{8\times 8} & per\ (stage, size) = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ (stage, size) = (0,2)\ oppure\ (1,2) \\ (H_2)_8 & per\ (stage, size)\ (1,0) \\ \begin{bmatrix} (H_1)_4 & 0 \\ 0 & (H_1)_4 \end{bmatrix} & per\ (stage, size)\ (1,1) \end{cases}, \tag{50}$$

$$B = \begin{cases} 2C_8^3 \times I_{8\times 8} & per\ (stage, size) = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ (stage, size) = (0,2)\ oppure\ (1,2) \\ -(H_6)_8 & per\ (stage, size)\ (1,0) \\ \begin{bmatrix} -(H_3)_4 & 0 \\ 0 & -(H_3)_4 \end{bmatrix} & per\ (stage, size)\ (1,1) \end{cases}, \tag{51}$$

$$G = \begin{cases} 2C_4^1 \times I_{8\times 8} & per\ (stage, size) = (0,0)\ oppure\ (0,1) \\ I_{8\times 8} & per\ (stage, size) = (0,2)\ oppure\ (1,2) \\ (H_4)_8 & per\ (stage, size)\ (1,0) \\ \begin{bmatrix} (H_2)_4 & 0 \\ 0 & (H_2)_4 \end{bmatrix} & per\ (stage, size)\ (1,1) \end{cases}, \tag{52}$$

$$D = \begin{cases} 2C_{16}^1 \times I_{8\times 8} & per\ stage = 0 \\ (H_1)_8 & per\ stage = 1 \end{cases}, \tag{53}$$

$$E = \begin{cases} 2C_{16}^{3} \times I_{8\times 8} & per\ stage = 0 \\ (H_5)_8 & per\ stage = 1 \end{cases}, \quad (54)$$

$$F = \begin{cases} 2C_{16}^{7} \times I_{8\times 8} & per\ stage = 0 \\ -(H_7)_8 & per\ stage = 1 \end{cases}, \quad (55)$$

$$G = \begin{cases} 2C_{16}^{5} \times I_{8\times 8} & per\ stage = 0 \\ -(H_3)_8 & per\ stage = 1 \end{cases}, \quad (56)$$

Figura 15: Schema dei blocchi che implementano la fase PROCESS

Blocchi che compongono la fase PROCESS per l'architettura scalabile. Il blocco QA, per esempio, riceve in ingresso due vettori a otto componenti (ogni componente è un pixel, che può aver già subito elaborazioni) e invia come uscite due vettori a otto componenti: il primo vettore è la somma dei due vettori in ingresso, mentre il secondo vettore è la differenza fra i due vettori in ingresso successivamente elaborata tramite l'operatore lineare A. Si noti che gli operatori A,B,C,D,E,F,G sono matrici 8x8.

Figura 16: Schema dettagliato del blocco QA

In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici (sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base ai valori delle variabili di controllo stage e size, seleziona un solo risultato, che costituisce una componente del vettore in uscita.

Figura 17: Schema dettagliato del blocco QB

In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici (sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base ai valori delle variabili di controllo stage

e size, seleziona un solo risultato, che costituisce una componente del vettore in uscita.

Figura 18: Schema dettagliato del blocco QC

In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici (sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base ai valori delle variabili di controllo stage

e size, seleziona un solo risultato, che costituisce una componente del vettore in uscita.

Figura 19: Schema dettagliato del blocco QD

In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici (sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base al valore della variabile di controllo stage, seleziona un solo risultato, che costituisce una componente del vettore in uscita.

Figura 20: Schema dettagliato del blocco QE

In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici (sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base al valore della variabile di controllo stage, seleziona un solo risultato, che costituisce una componente del vettore in uscita.

Figura 21: Schema dettagliato del blocco QF. In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici (sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base al valore della variabile di controllo {\em stage}, seleziona un solo risultato, che costituisce una

componente del vettore in uscita.

Figura 22: Schema dettagliato del blocco QG. In questo schema sono mostrate in dettaglio le singole componenti dei due vettori in ingresso e gli operatori aritmetici(sommatori ecc.) che agiscono su ciascuna componente. I risultati vengono inviati ai MUX (sulla destra), ciascuno dei quali, in base al valore della variabile di controllo *stage*, seleziona un solo risultato, che costituisce una componente del vettore in uscita.

Figura 23: Fase ORDER per l'architettura scalabile. Gli ingressi sono i 64 pixel dopo essere stati elaborati nella fase PROCESS.

L'ordinamento degli ingressi viene operato per mezzo di MUX controllati dalla variabile *size*.

Le 64 uscite sono le componenti degli otto vettori a otto componenti ***l'***, ***m'***, .., ***s'***.

Figura 24: Fase OUTPUTper l'architettura scalabile.Gli ingressi sono i 64 coefficienti 2-D DCT.

L'ordinamento degli ingressi viene operato per mezzo di MUX controllati dalla variabile *size*.

Le 64 uscite sono i pixel che costituiscono il blocco di uscita.

RIVENDICAZIONI

1. Metodo di calcolo della trasformata coseno discreta (DCT) per blocchi di pixels di un'immagine, caratterizzato dal fatto che comprende le operazioni di definire primi blocchi di suddivisione di detta immagine denominati blocchi *range*, aventi una dimensione frazionaria e scalabile di $N/2^i * N/2^i$, dove i è un numero intero, rispetto ad una dimensione massima predefinita di un blocco di $N*N$ pixels, denominato blocco *domain*, traslabile per intervalli di $N/2^i$ pixels, ed effettuare il calcolo della DCT parallelamente su 2^i blocchi *range* di suddivisione di un blocco *domain* di $N*N$ pixels.

2. Metodo secondo la rivendicazione 1, caratterizzato dal fatto che il calcolo della DCT parallelamente su tutti i blocchi *range* di suddivisione di un certo blocco *domain* è eseguito mediante una struttura hardware e comprende le seguenti operazioni:

a) ordinare i pixels in funzione della suddivisione in blocchi range di una certa dimensione riordimando i pixels di ingresso in un numero 2^i di sequenze o vettori di 2^i componenti;

b) calcolare in parallelo 2^i DCT monodimensionali elaborando detti vettori definiti nella precedente fase a);

c) ordinare le sequenze di uscita dei dati monodimensionali DCT relativi a detti 2^i vettori;

d) completare il calcolo parallelo di detti 2^i DCT bidimensionali elaborando dette sequenze di uscita prodotte nella fase c);

e) ordinare le sequenze di uscita prodotte nella fase d) in un numero 2^i di vettori di coefficienti DCT bidimensionali.

3. Metodo secondo la rivendicazione 2, caratterizzato dal fatto che il calcolo parallelo di dette 2^i DCT monodimensionali della fase b) e il

completamento del calcolo parallelo di dette 2^i DCT bidimensionali della fase d) sono effettuati suddividendo le sequenze risultanti rispettivamente dalla fase a) e dalla fase c) in gruppi di quattro elementi scalari, calcolandone le relative somme e differenze mediante sommatori e sottrattori e moltiplicando i risultati somma e differenza per rispettivi coefficienti in modo reiterativo fino a completare il calcolo dei relativi quattro coefficienti DCT, rispettivamente monodimensionali e bidimensionali.

4. Metodo di compressione dei dati di un'immagine, da memorizzare o trasmettere, mediante codifica frattale, caratterizzato dal fatto che la trasformazione frattale è effettuata nel *domain* della trasformata coseno discreta (DCT) attraverso le seguenti operazioni

- suddividere un'immagine in detti due distinti tipi di blocchi di pixels;
- calcolare parallelamente la trasformata coseno discreta (DCT) di tutti i 2^i blocchi *range* e di un relativo blocco *domain*;
- classificare i blocchi *range* trasformati in funzione della loro relativa complessità rappresentata dalla somma dei valori della terna di coefficienti AC;
- applicare la trasformata frattale nel dominio della DCT ai dati dei blocchi *range* con classificazione di complessità superiore ad una soglia prestabilita e memorizzare solo il coefficiente DC dei blocchi *range* con complessità inferiore a detta soglia, identificando un relativo blocco *domain* di appartenenza del blocco *range* in trasformazione tale da produrre la migliore approssimazione frattale dello stesso blocco *range*;
- calcolare un'immagine differenza tra ciascun blocco *range* e la sua approssimazione frattale;
- quantizzare detta immagine differenza nel dominio della DCT impiegando una tabella di quantizzazione predisposta in funzione delle caratteristiche della vista umana;

- codificare l'immagine differenza quantizzata mediante un metodo di codifica basato sulla probabilità dei coefficienti di quantizzazione;

- memorizzare o trasmettere il codice di codifica per ciascun blocco *range* compresso nel dominio della DCT e il coefficiente DC di ciascun blocco *range* non compresso.

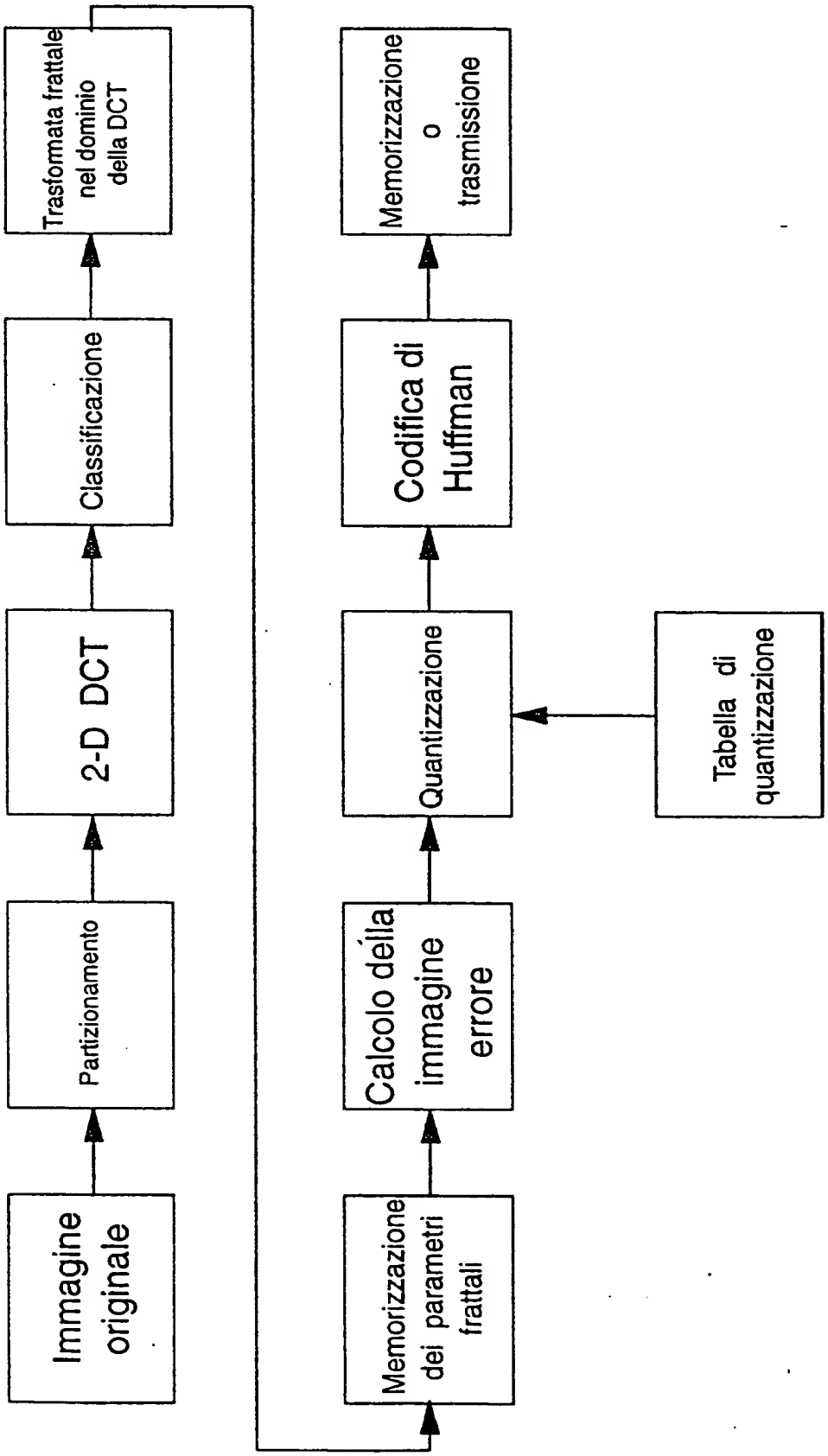

FIG. 1

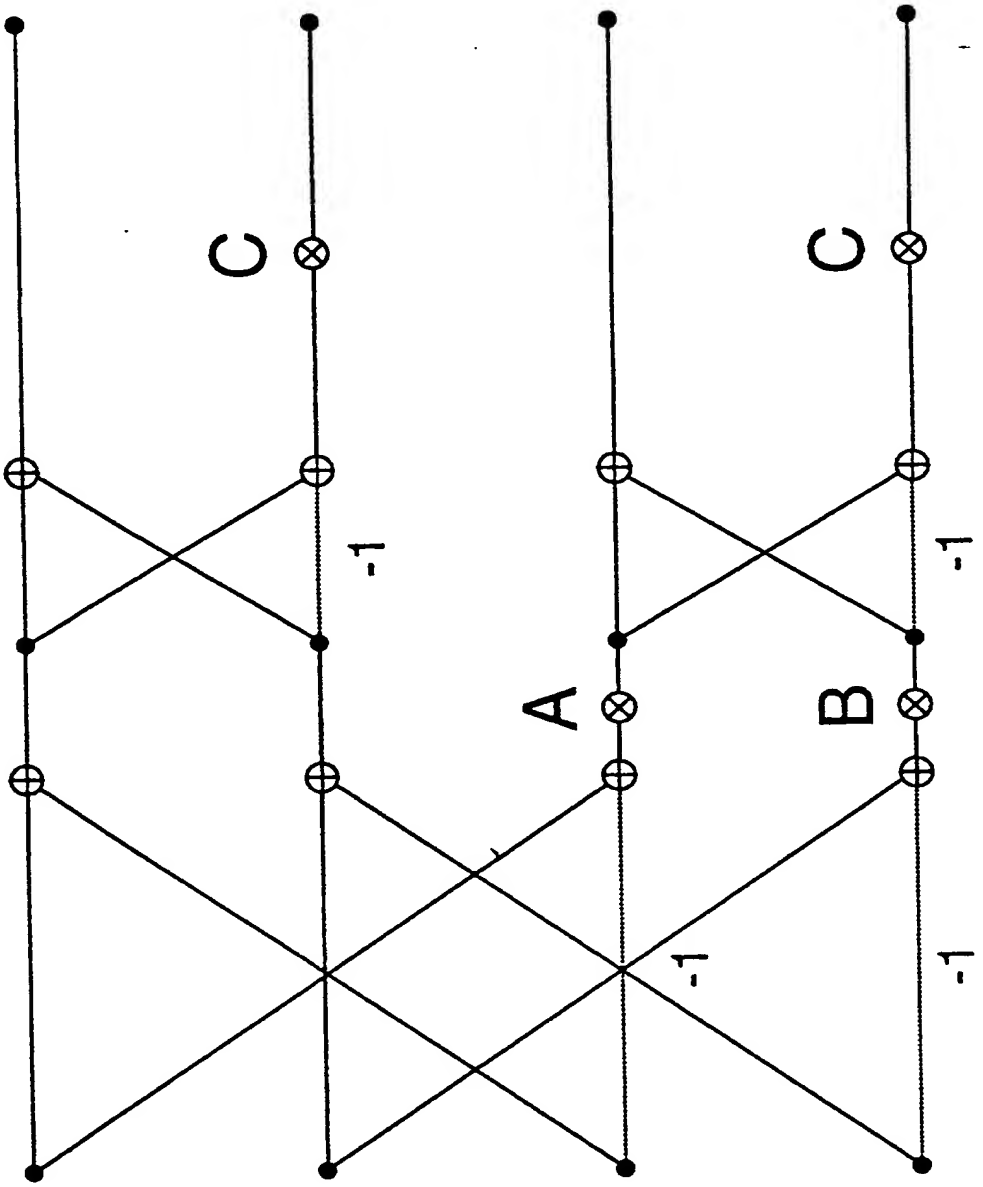

FIG. 2

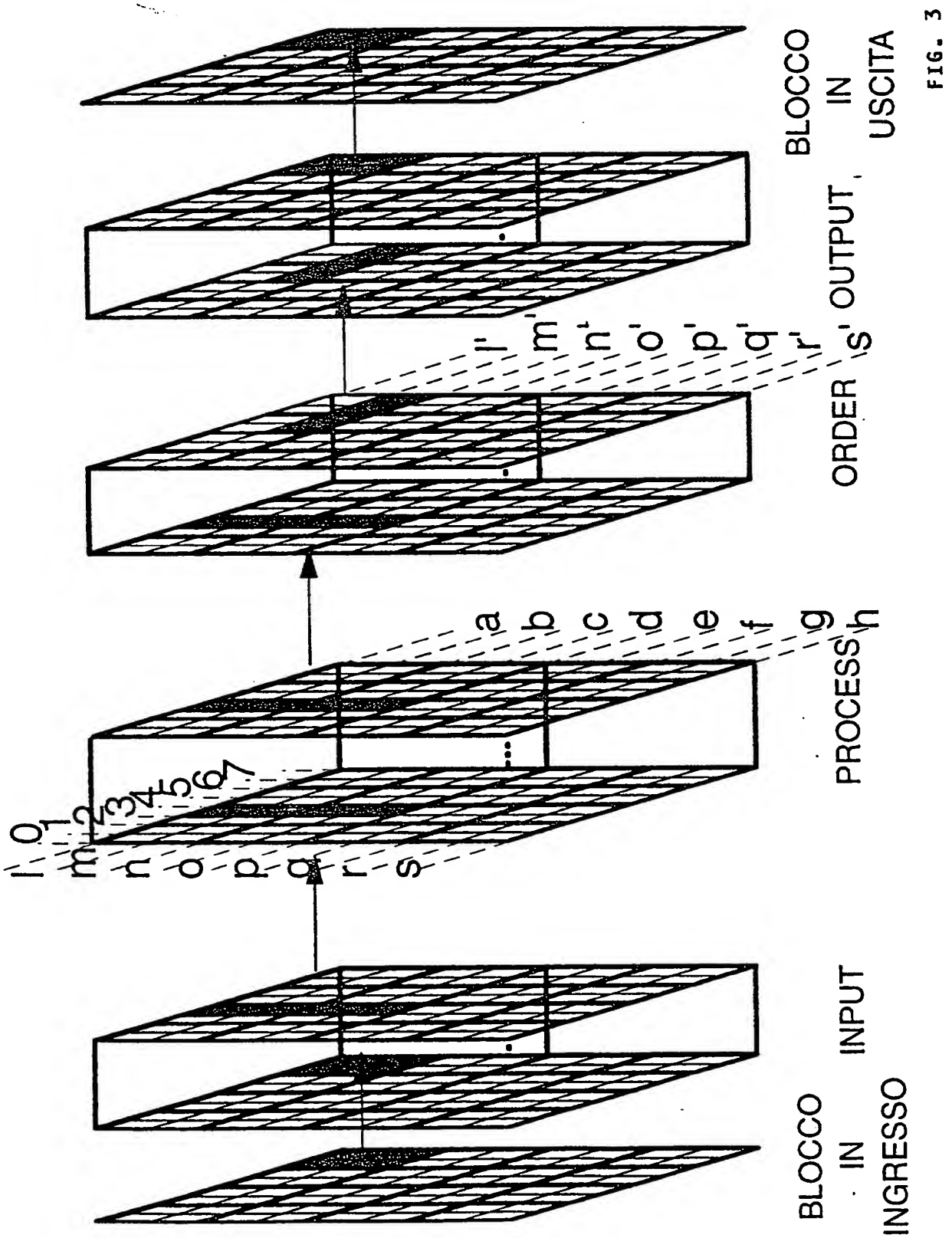

FIG. 3

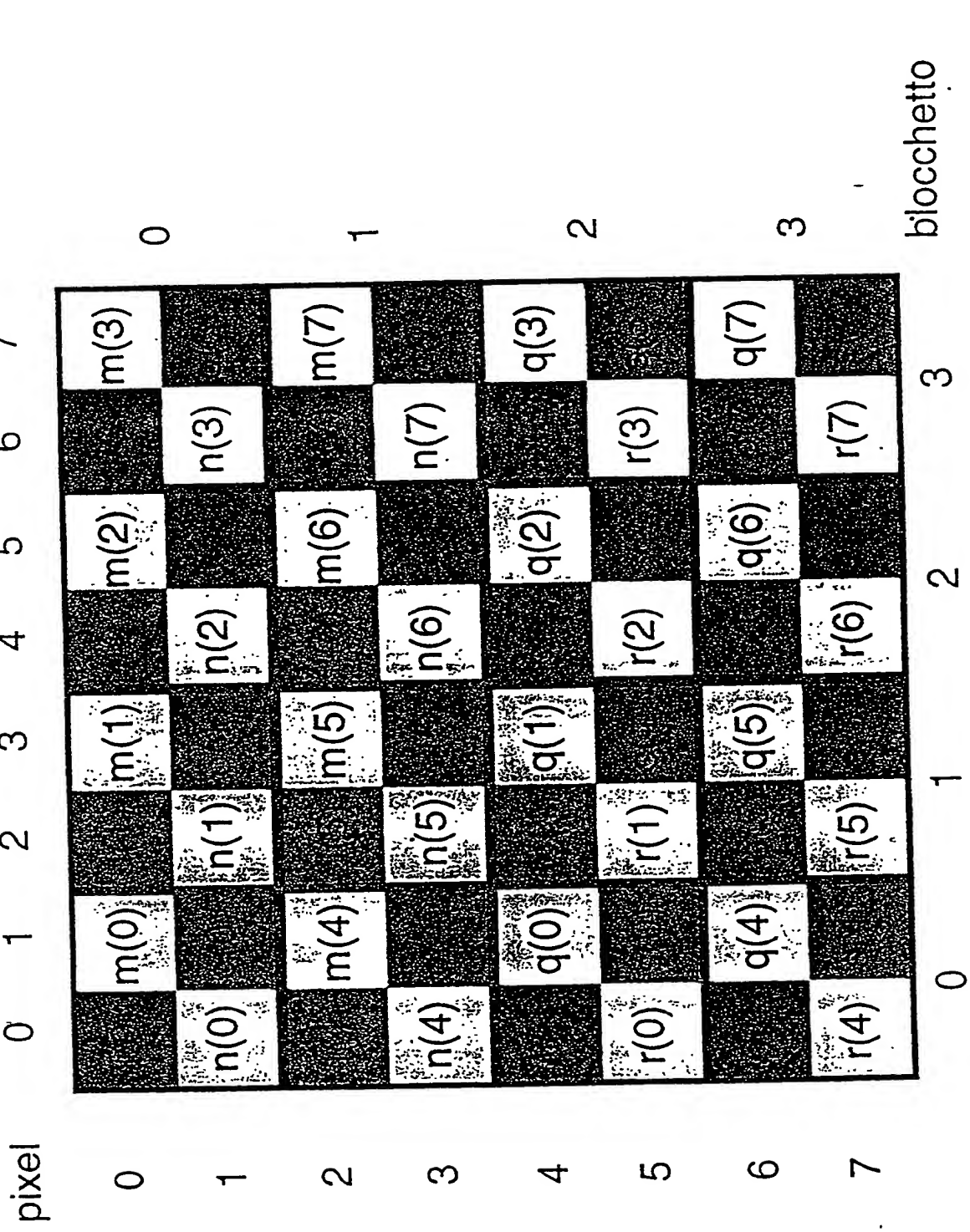

FIG.4

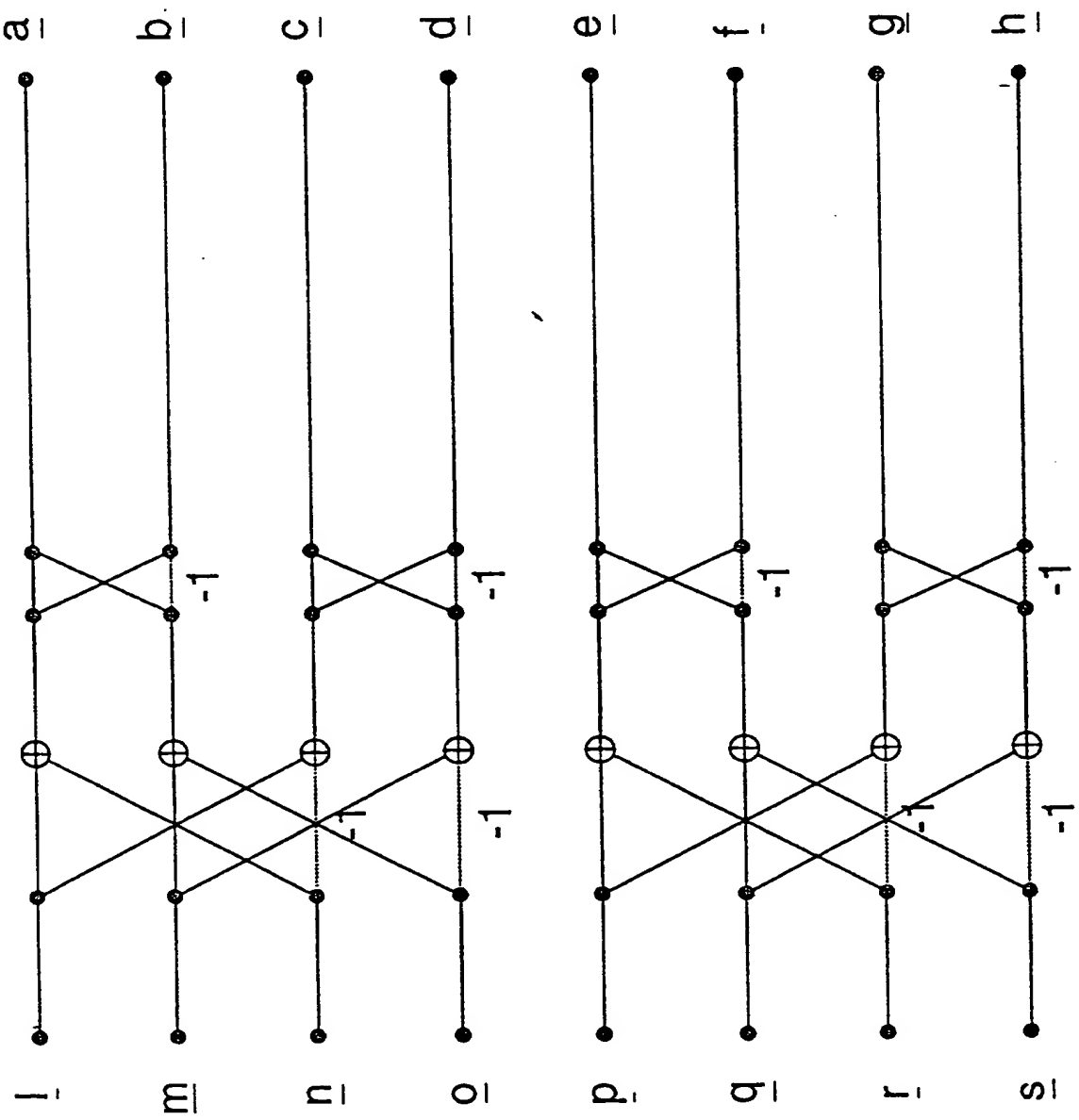

FIG. 5

6/24

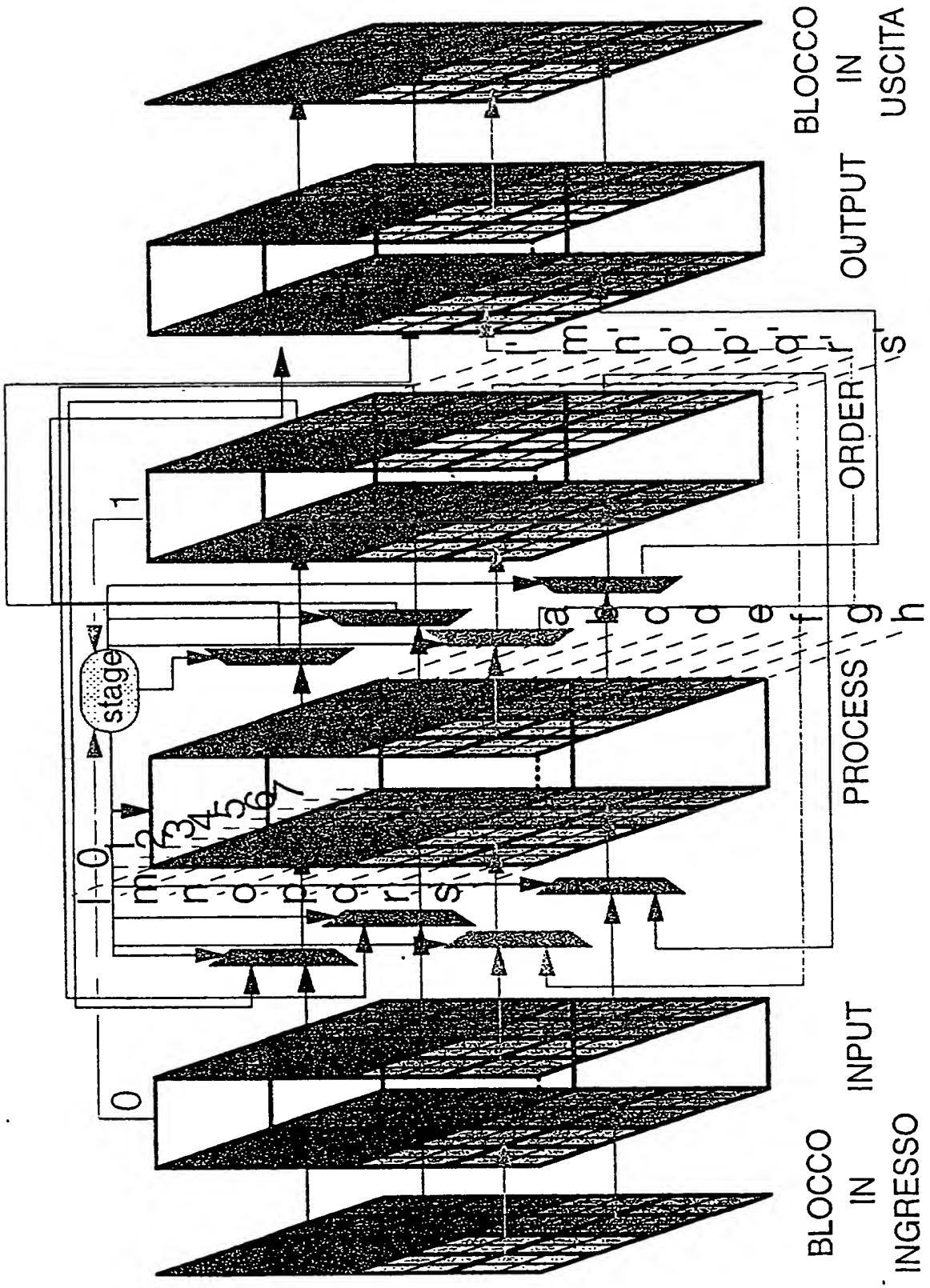

FIG. 6

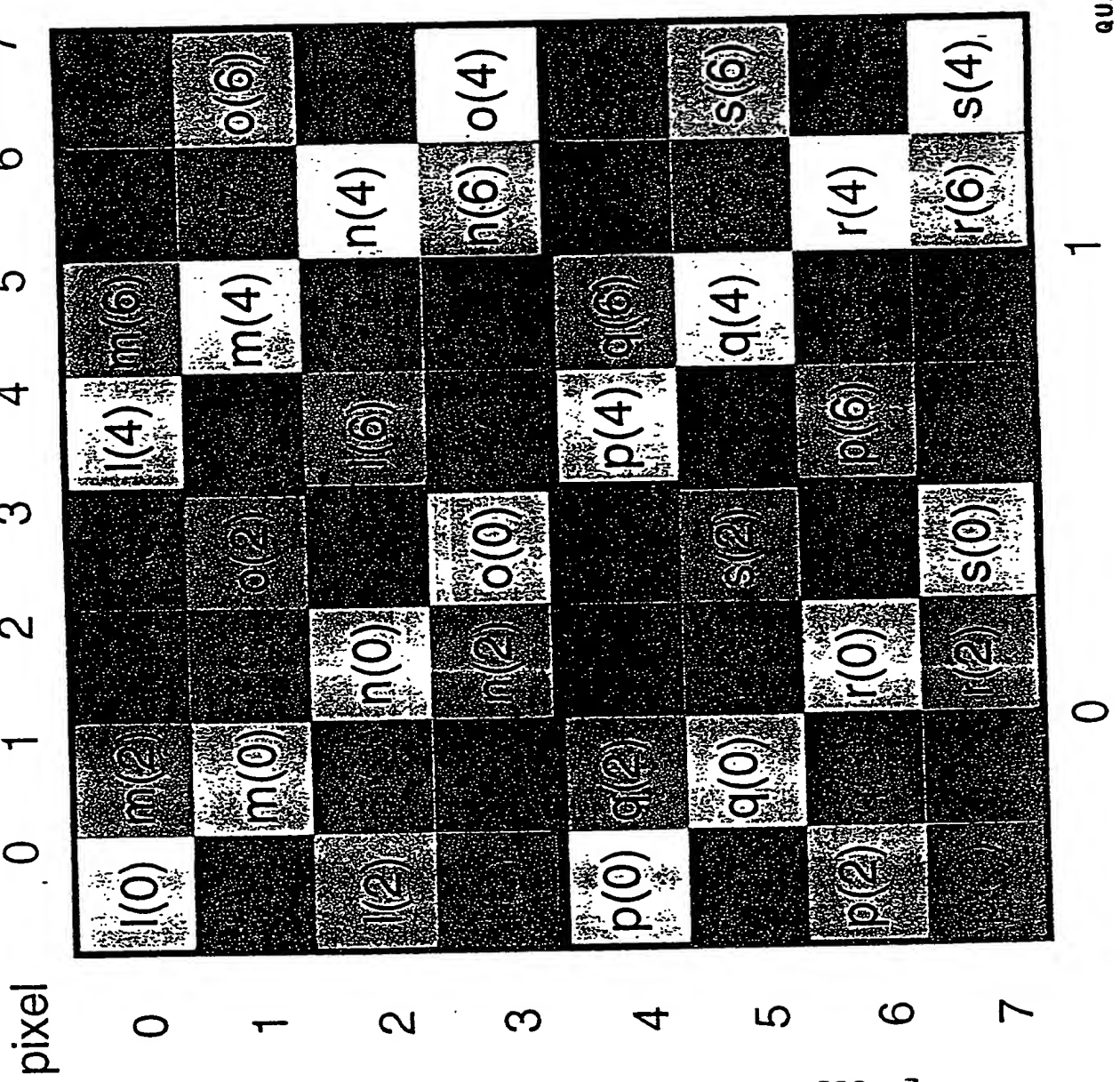

FIG. 7

DOCID: <E1 9883052204>

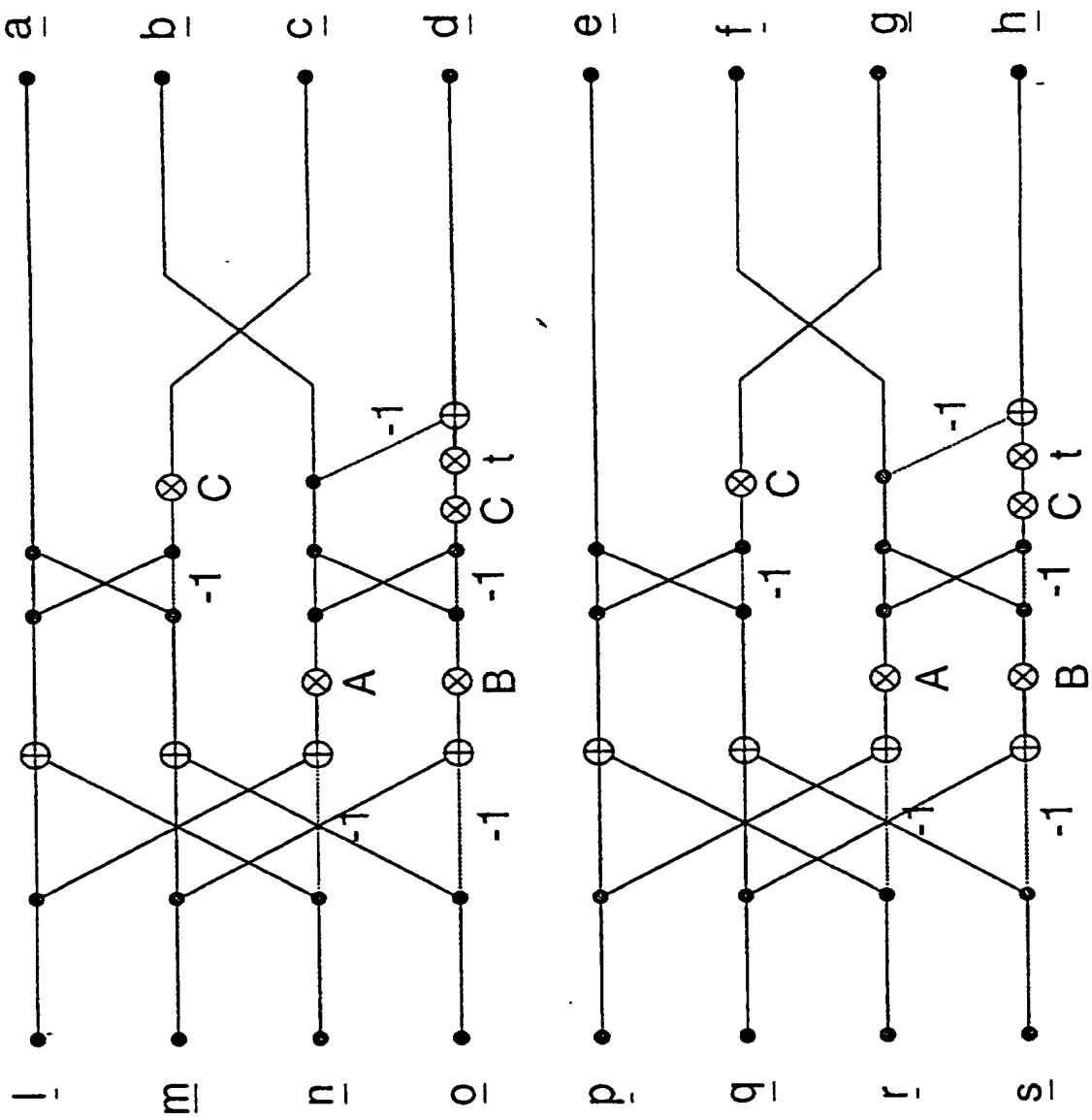

FIG. 8

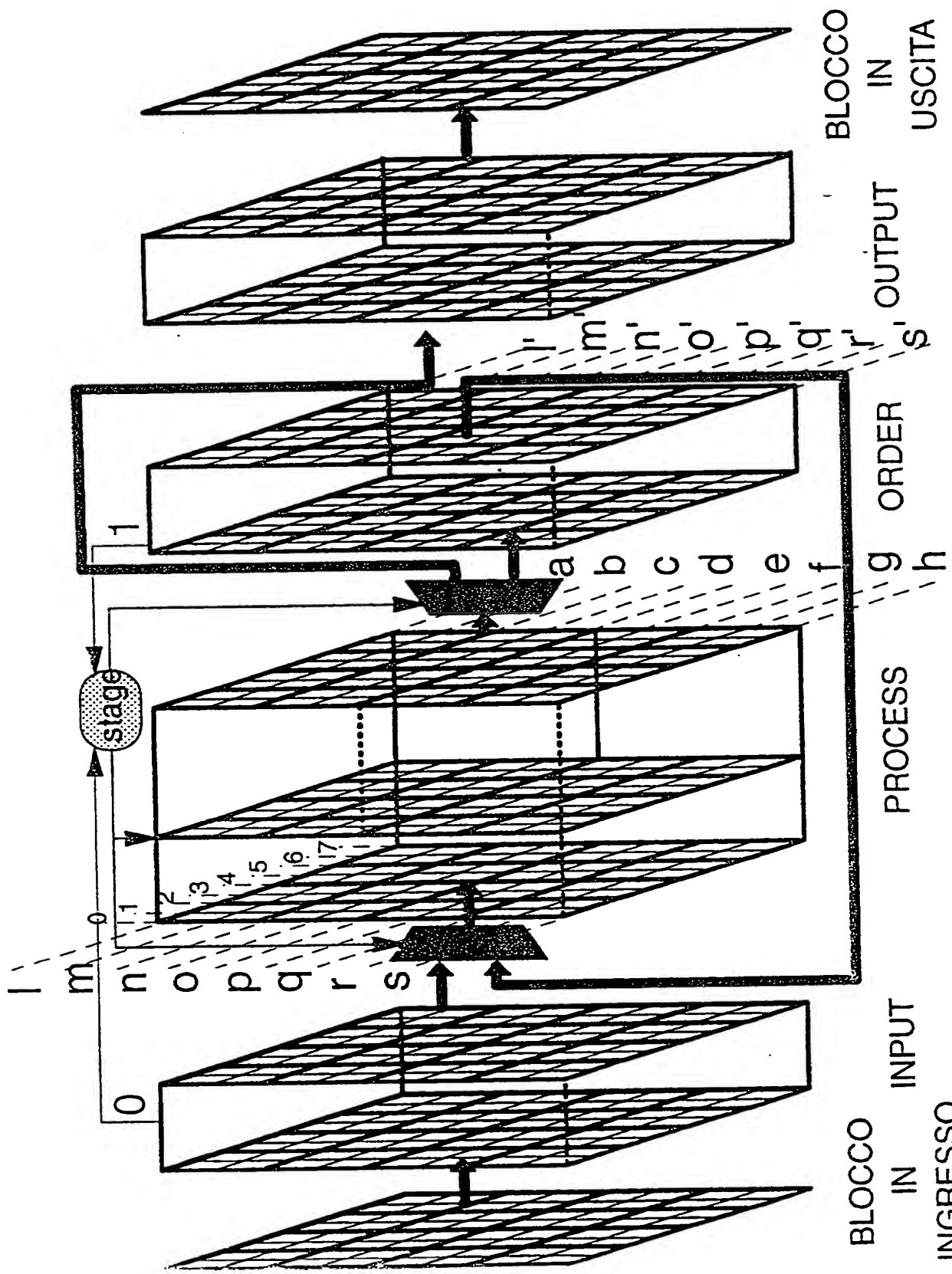

FIG. 9

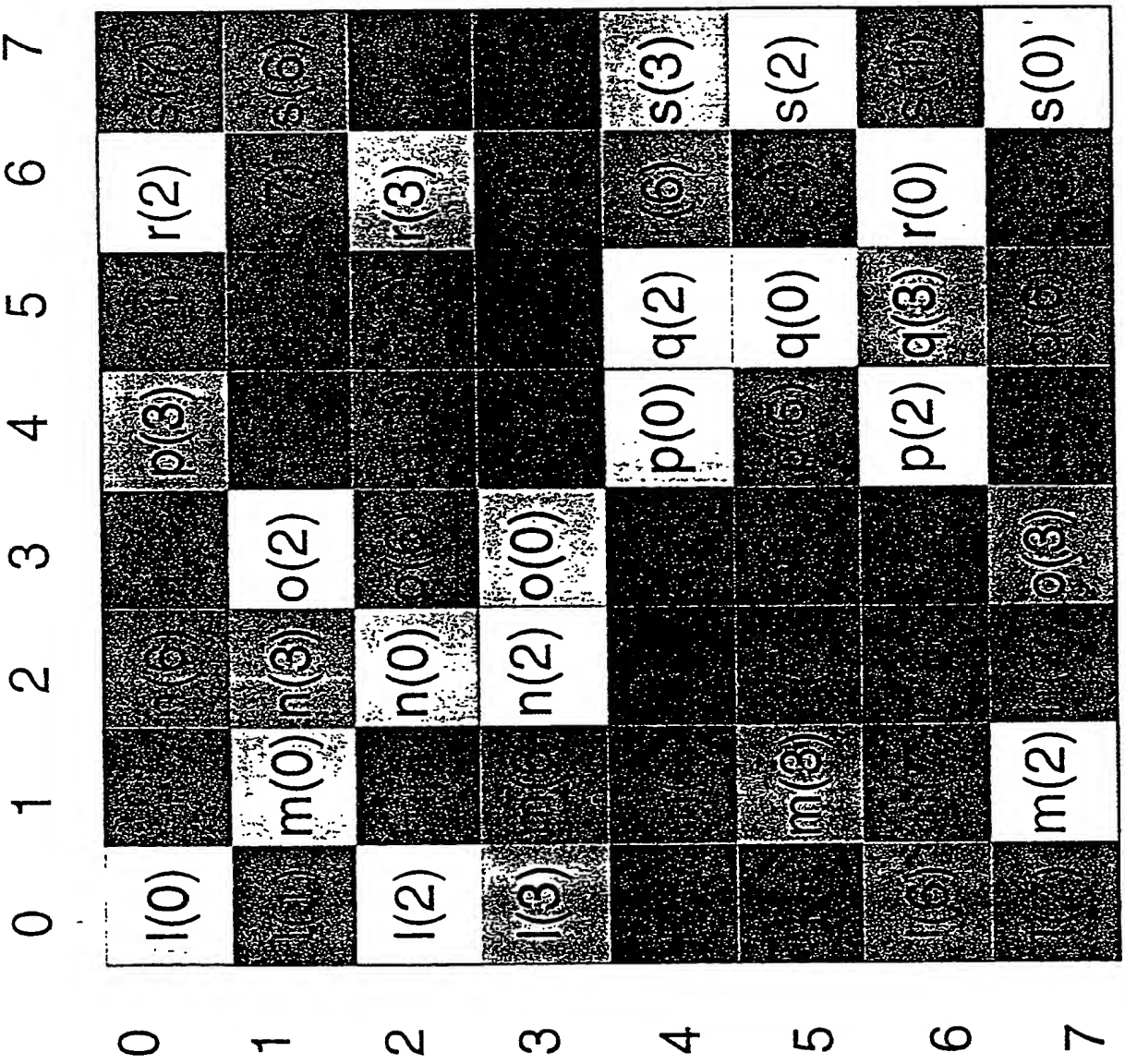

FIG. 10

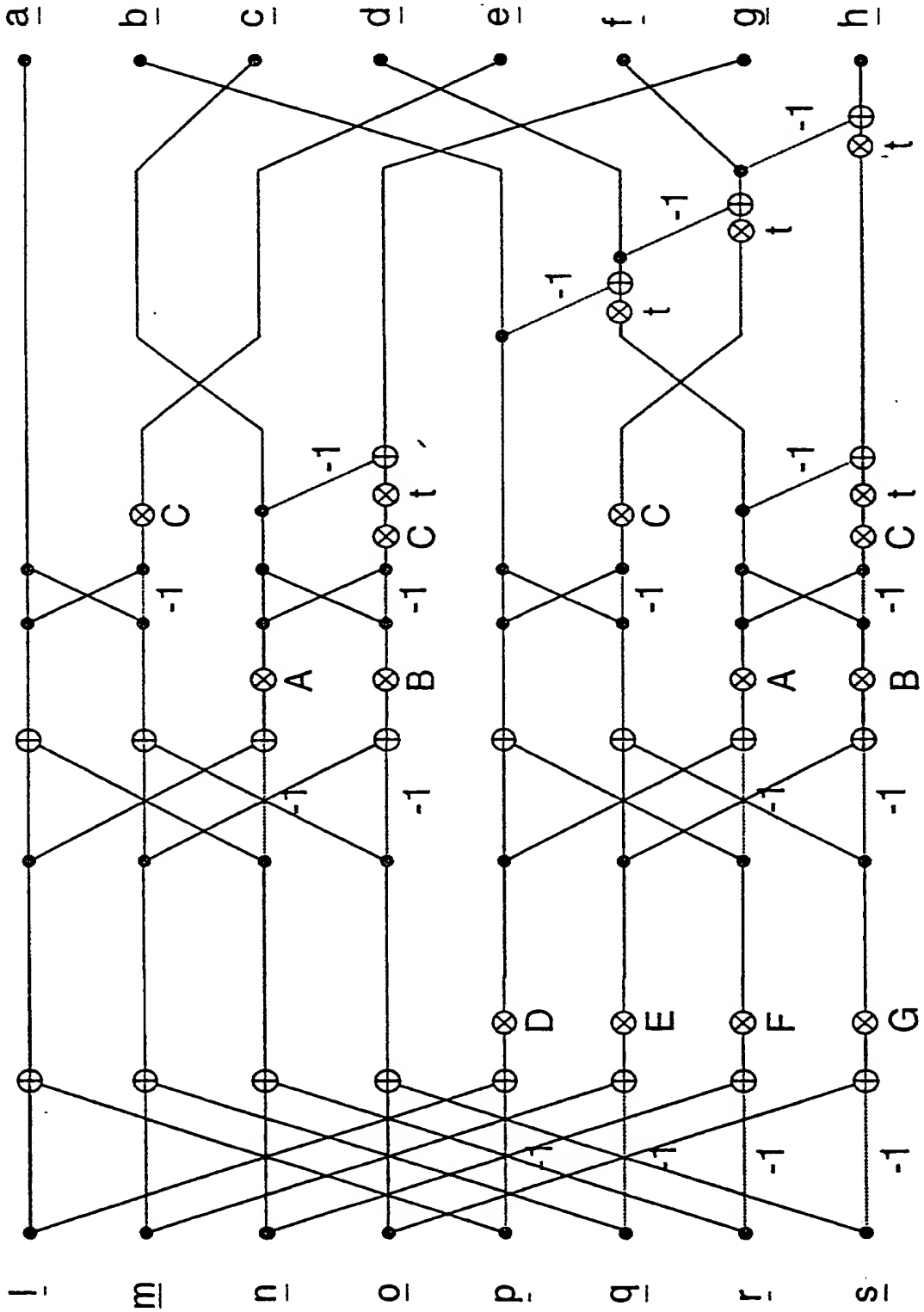

FIG. 11

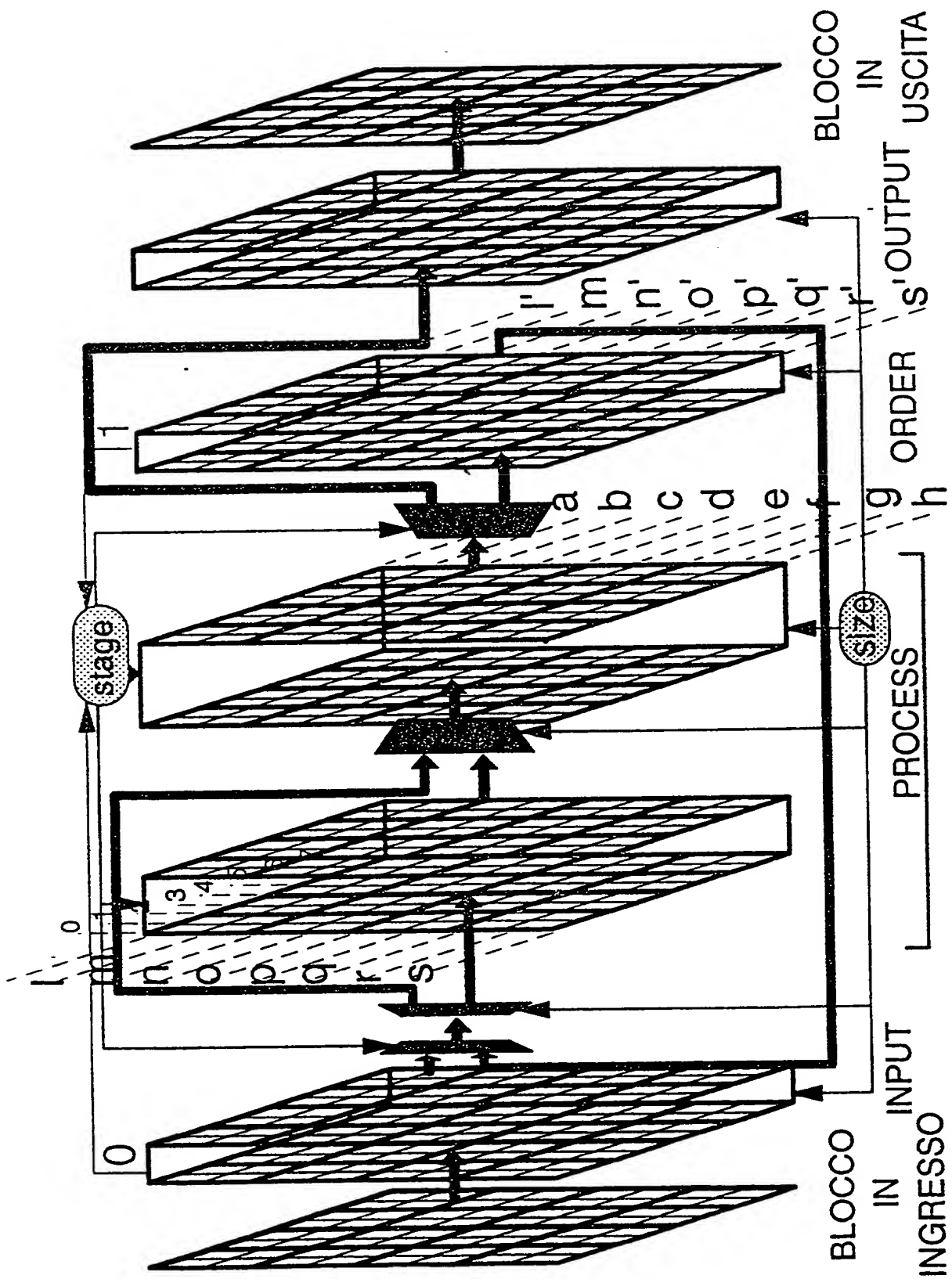

FIG. 12

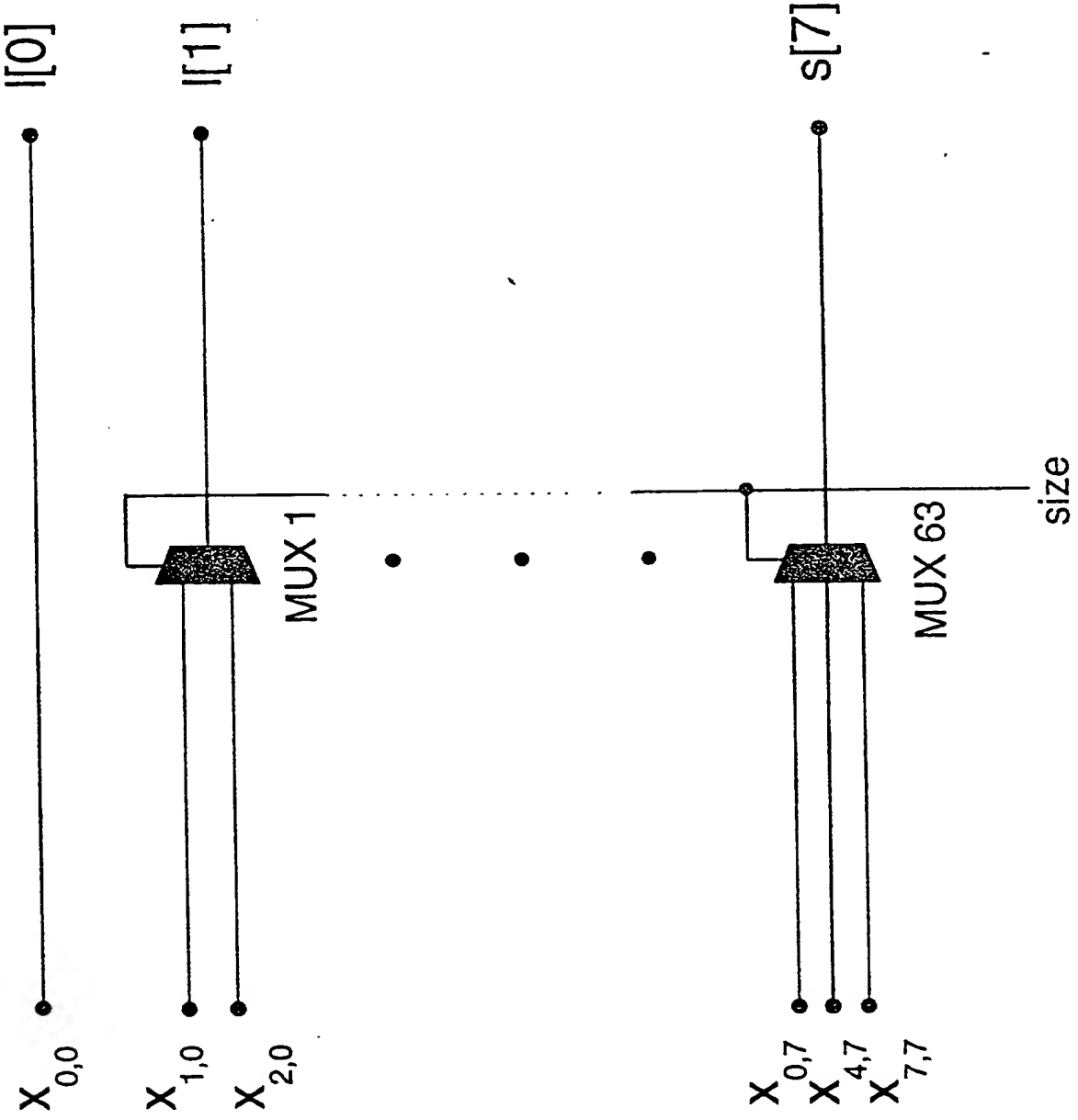

FIG. 13

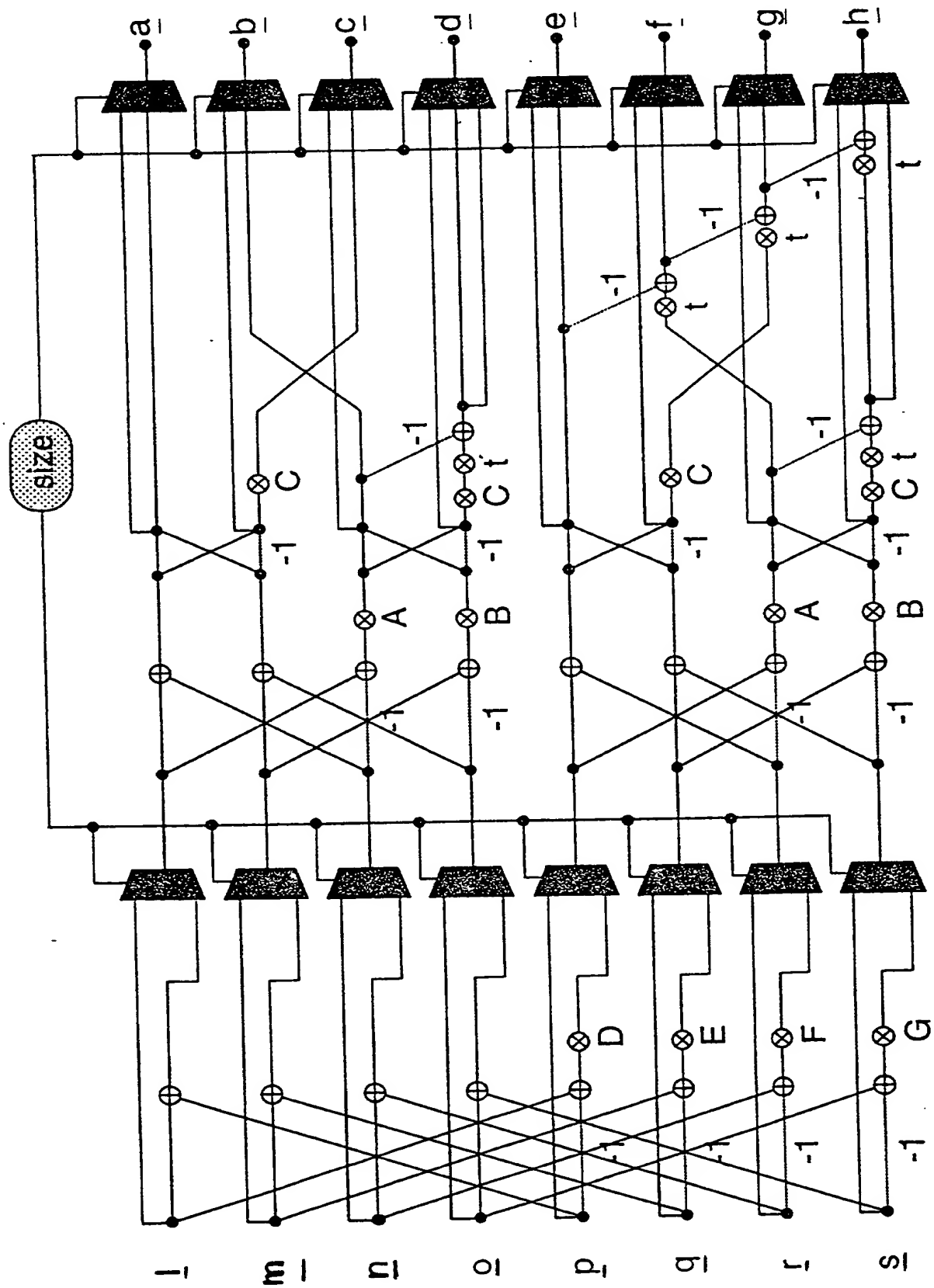

FIG. 14

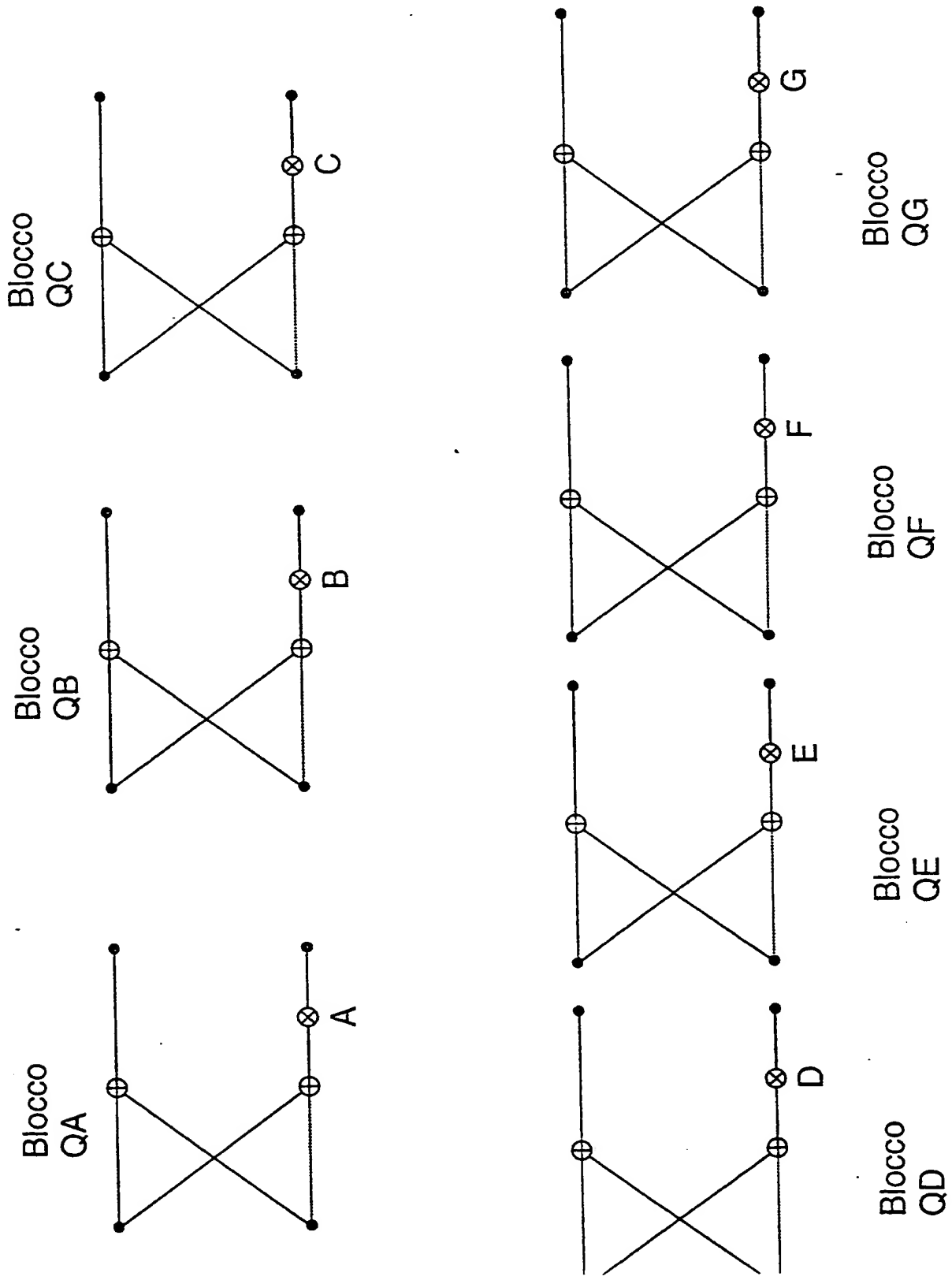

FIG. 15

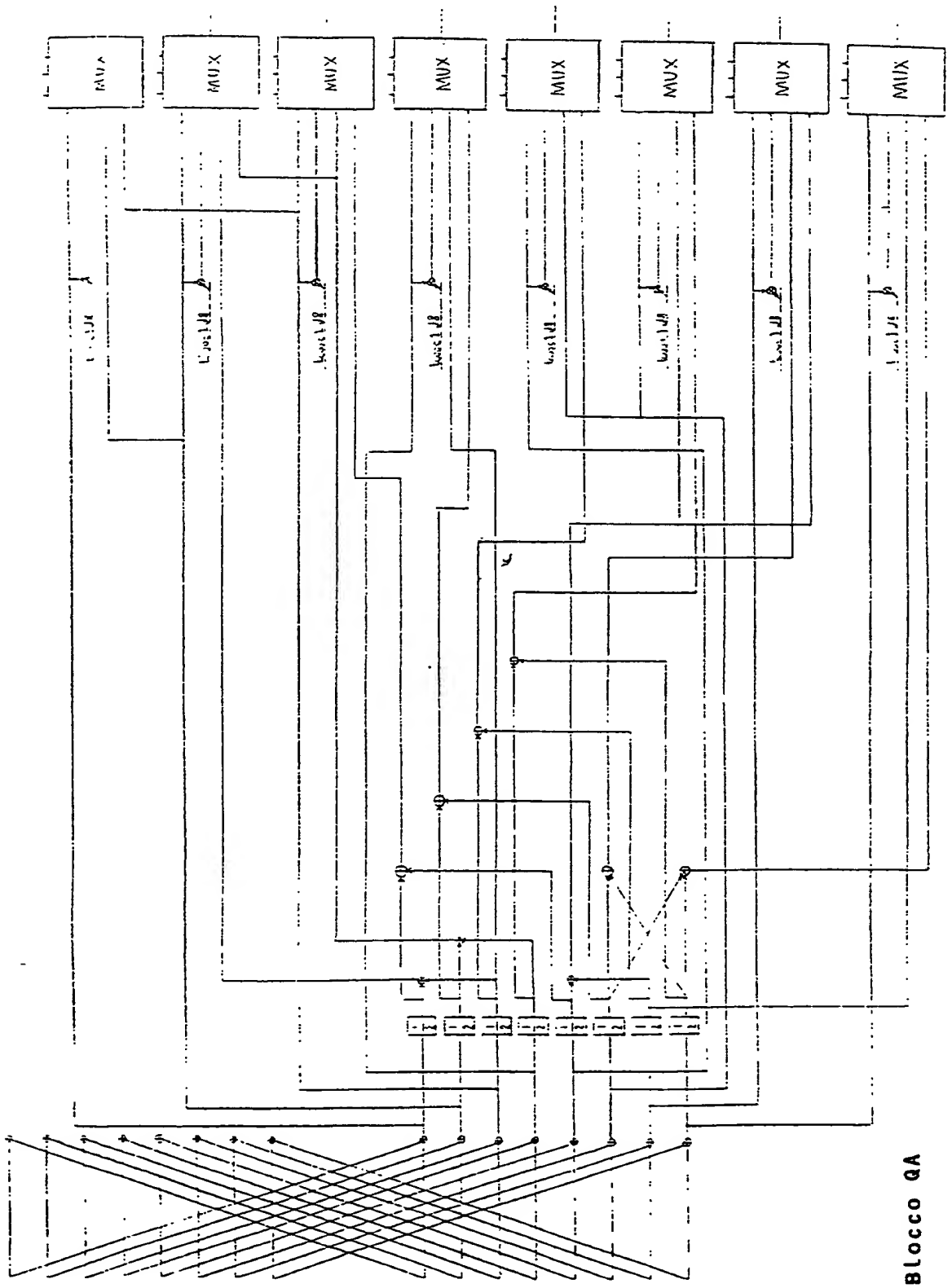

FIG. 16

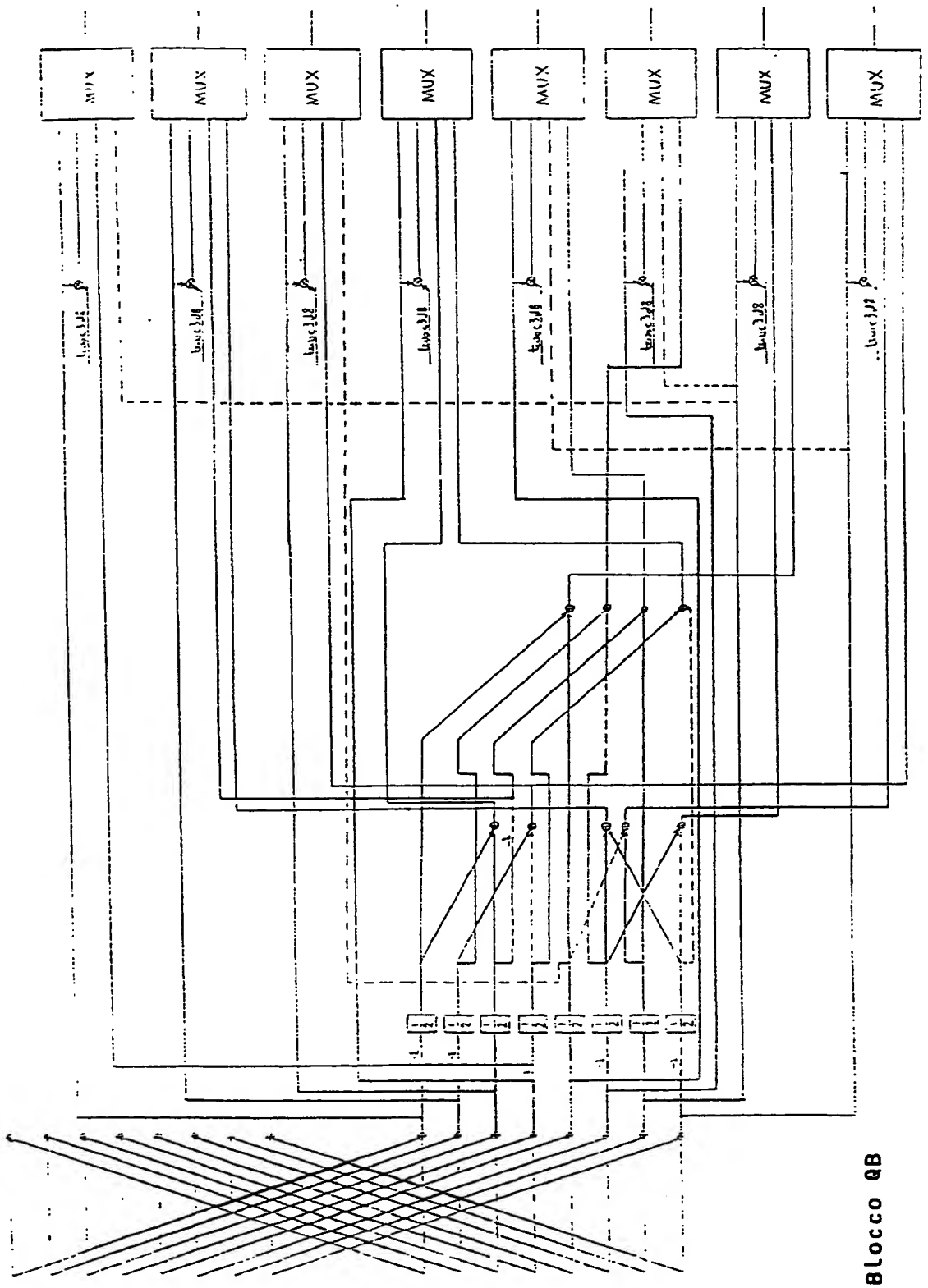

FIG. 17

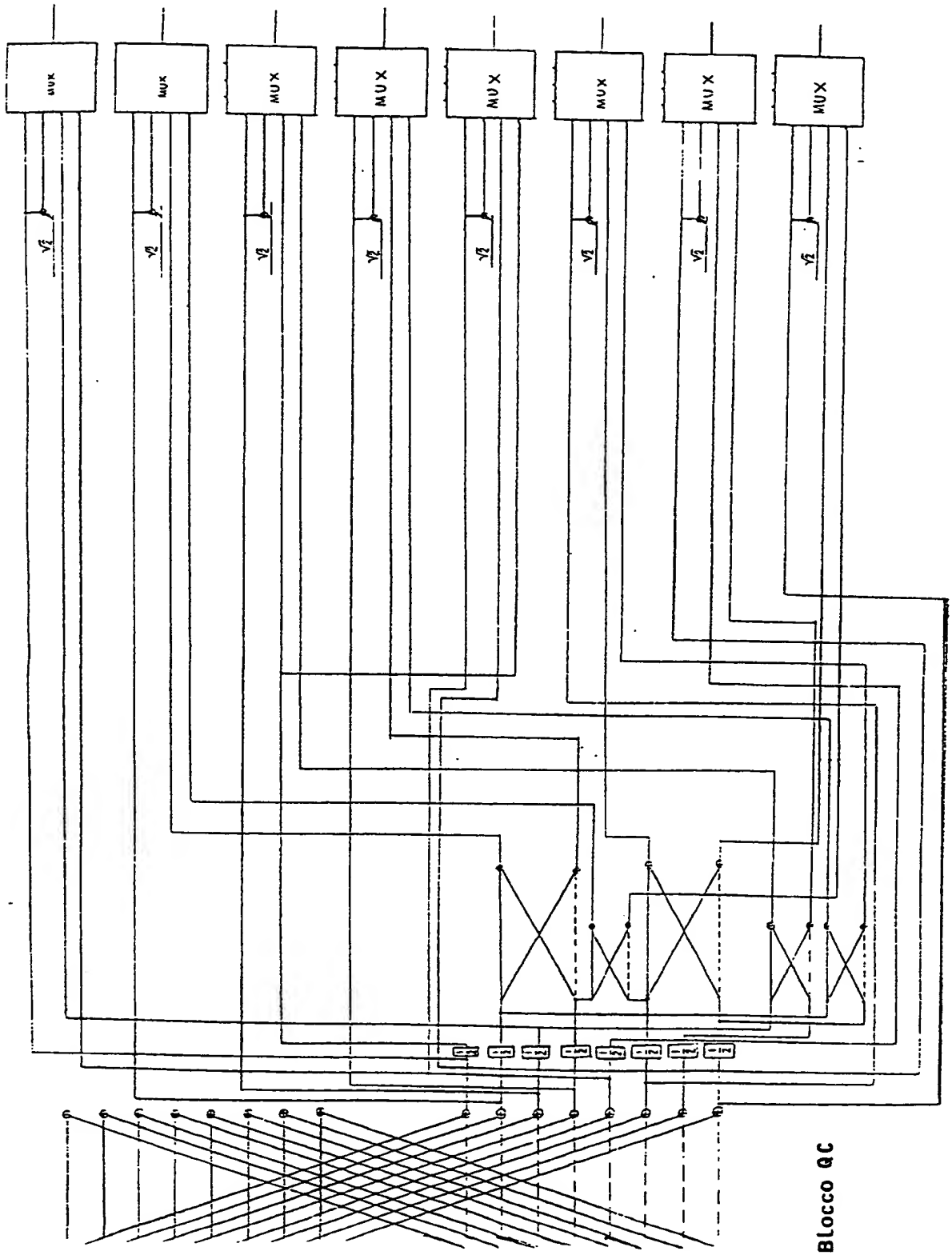

FIG. 18

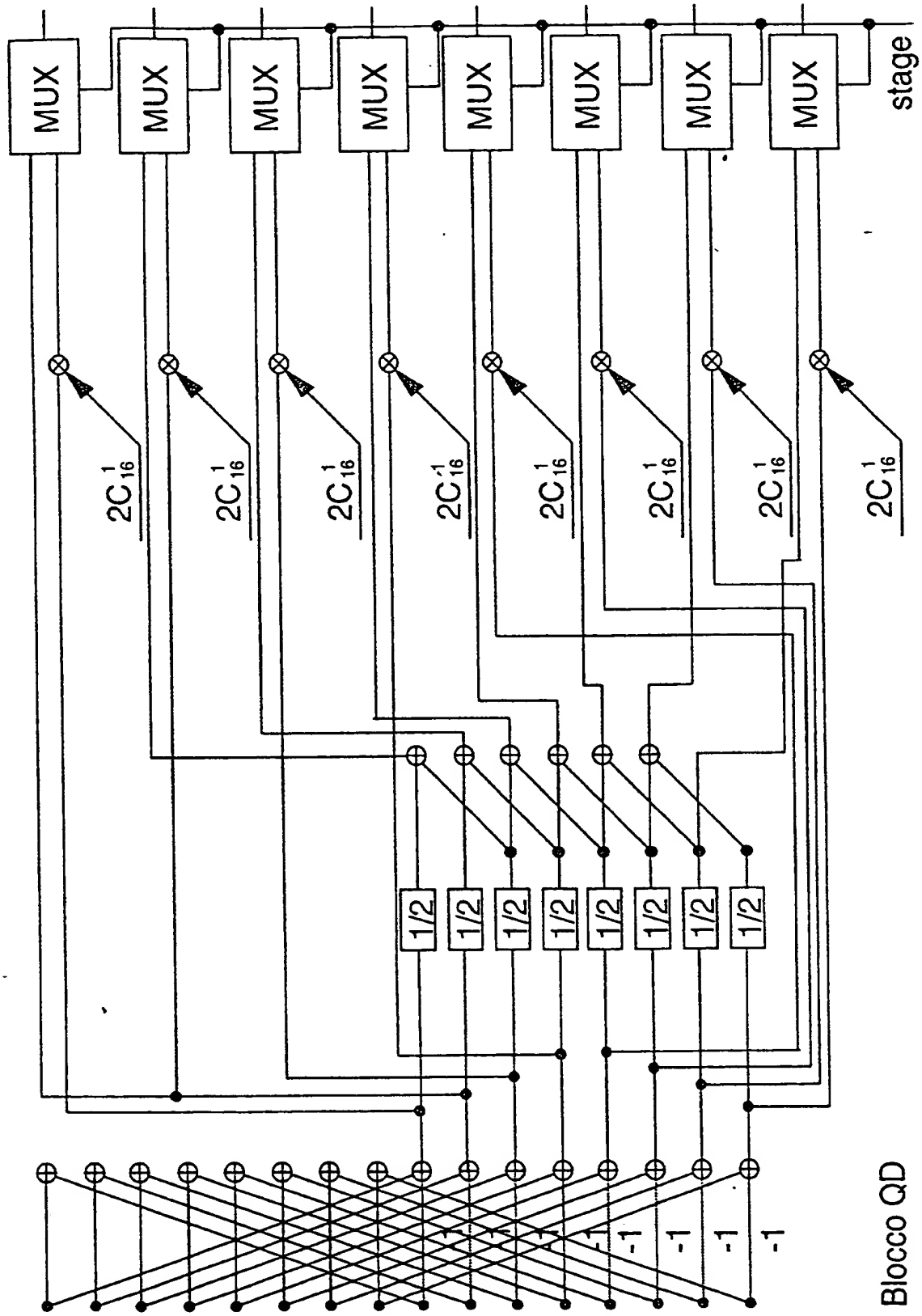

FIG. 19

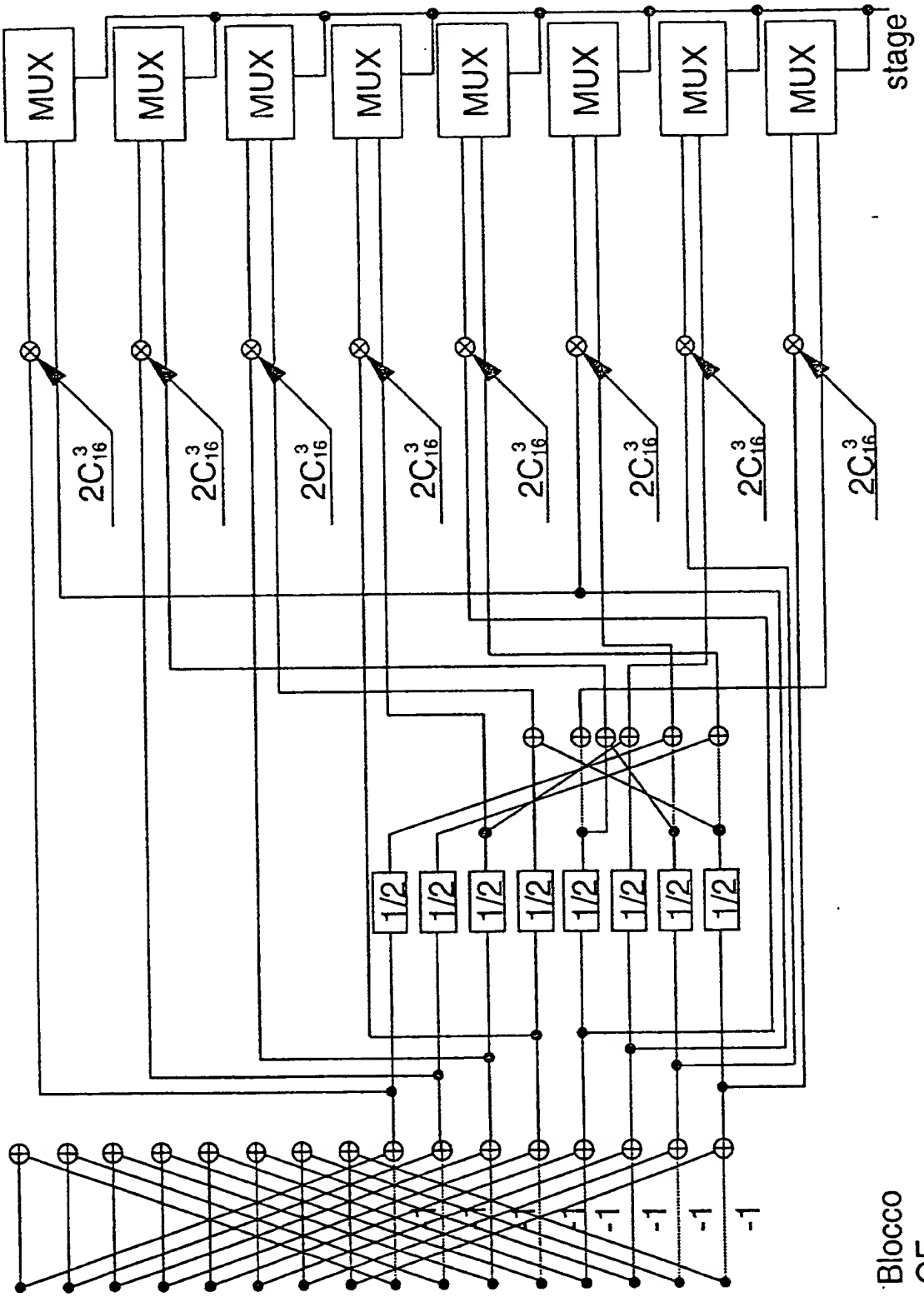

FIG. 20

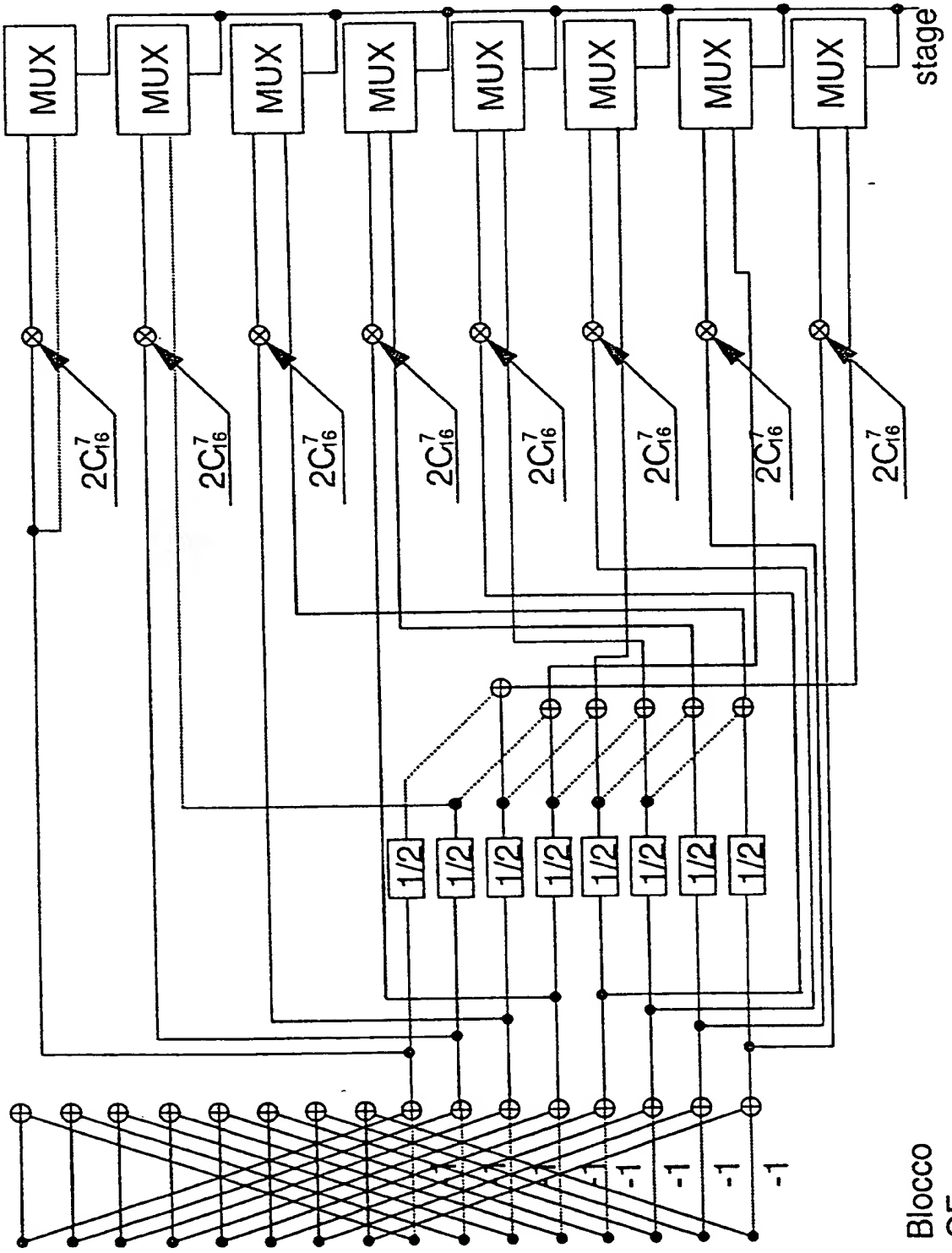

FIG. 21

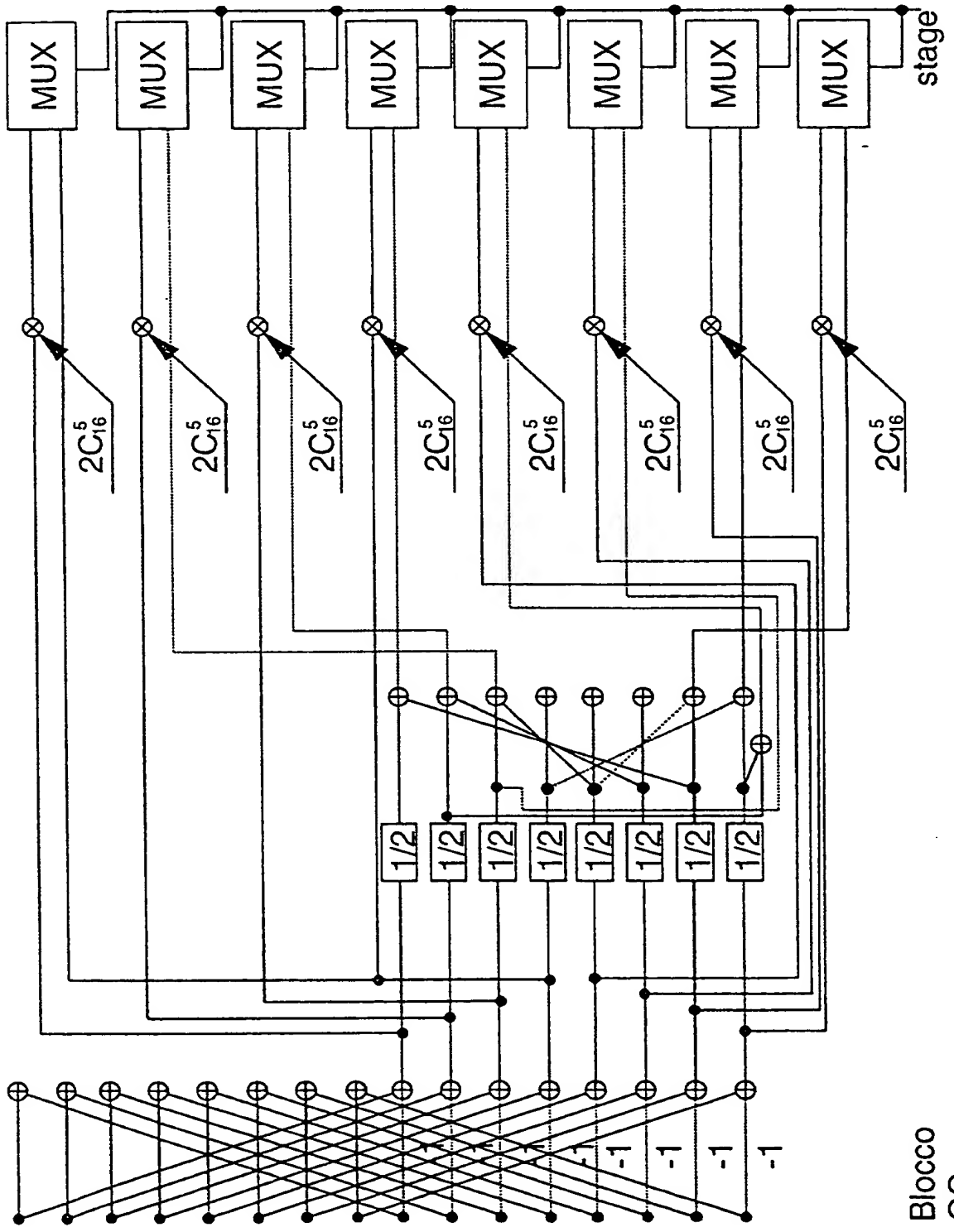

FIG. 22

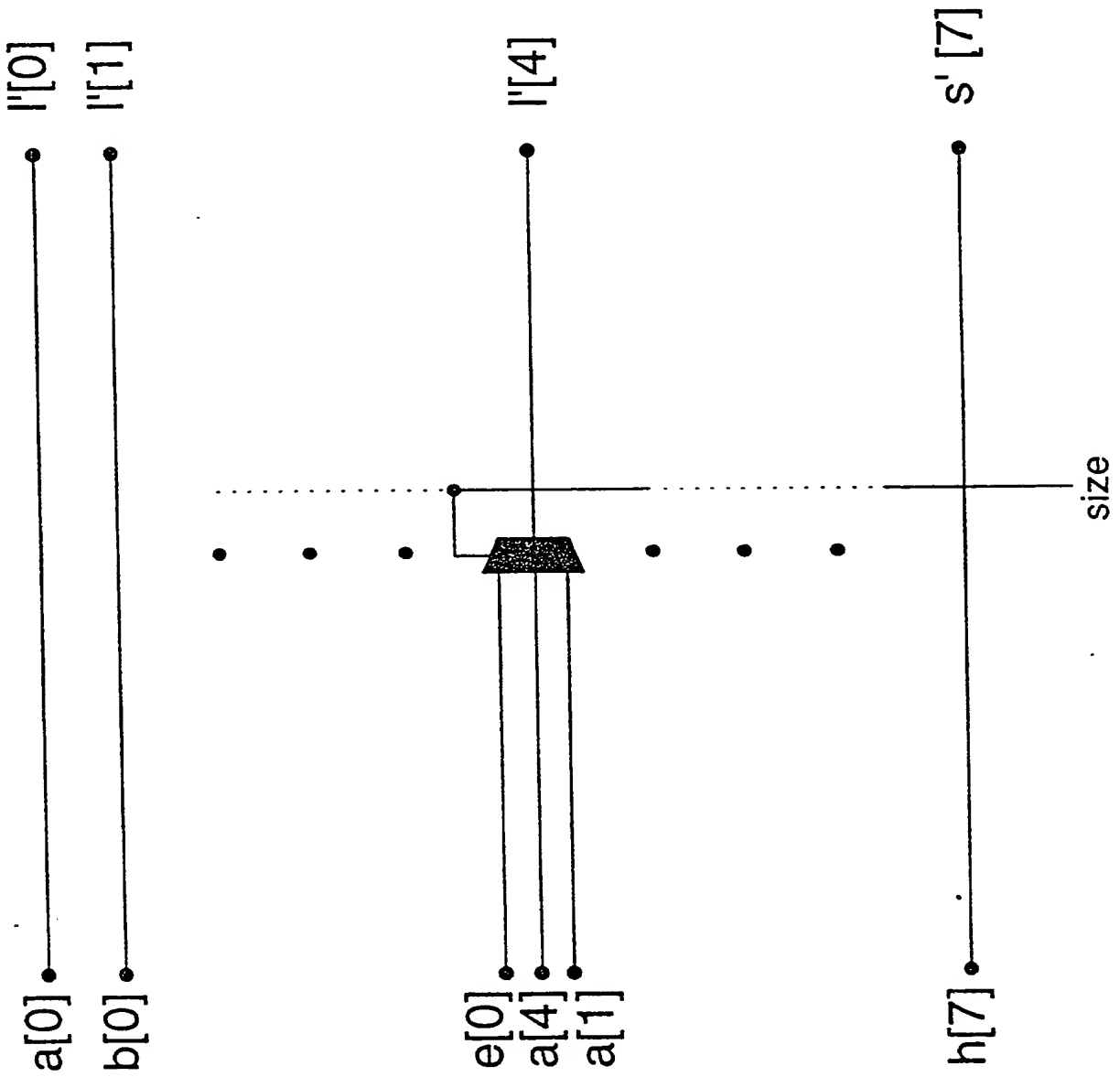

FIG. 23

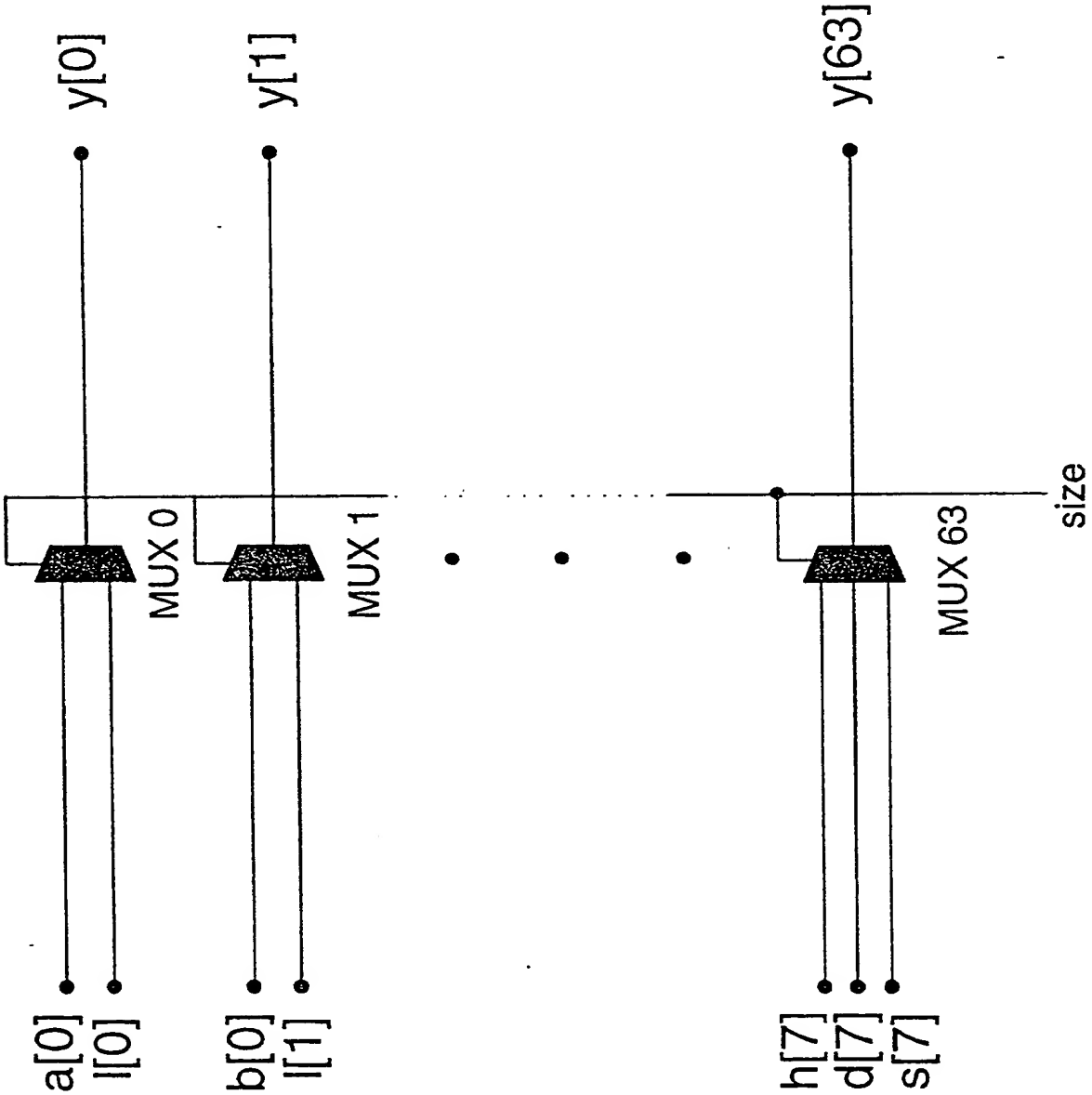

FIG. 24